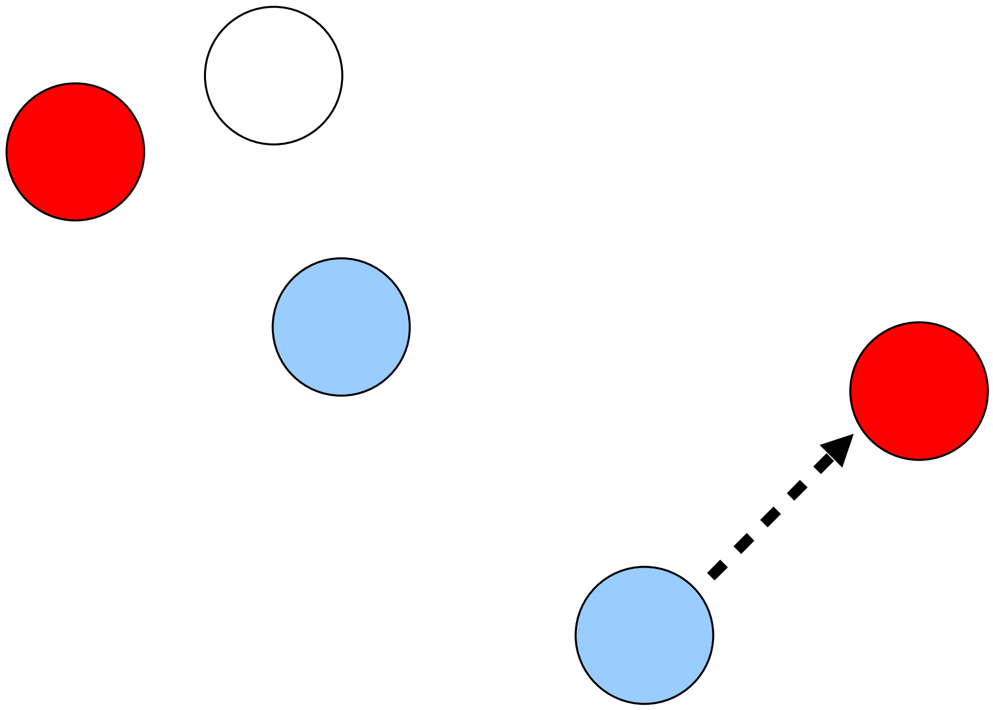
Taskposé: A Dynamic Task-Based Window Management Aid



Michael Bernstein
Stanford University, Symbolic Systems Program
Stanford, CA 94305

Advisor: Jeff Shrager, Symbolic Systems Program, Stanford University

Signed: Date:

Second Reader: Terry Winograd, Computer Science Department, Stanford University

Signed: Date:

Abstract

The window manager, a program which helps users organize and access their computers' open windows, is central to many aspects of computer work. Research in window managers has recently 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. However, our fieldwork and background theory suggest that neither is necessarily the case. Instead, we focus on *association* as an organizational scheme – windows can associate with tasks to varying degrees. We then introduce Taskposé, a prototype system that capitalizes on this idea through a full-screen graphical interface, and report on a weeklong user study. Finally, we comment on future directions for the prototype.

The Taskposé software is available for download on any Windows computer. Please contact the author at mbernst@stanfordalumni.org to obtain a copy.

Table of Contents

Introduction

Human activity is characterized by complex patterns in both the physical and digital worlds. Physical desks are nests of barely controlled chaos: reams of paper and documents, books and writing implements litter their surface. Most computer desktops are likewise dizzying arrays of open programs, files in various degrees of completion, and to-dos. Existing solutions to this problem will not scale. Users are now keeping eight or more windows open on their desktop 78% of the time (Hutchings, Smith, Meyers, Czerwinski, & Robertson, 2004), and this number is sure to rise as screen space and memory become cheaper and more aspects of our lives go online.

The physical desk shows signs of organization even in the face of challenging complexity: documents sorted into piles, Post-It notes indicating reminders or items of priority, and perhaps even a paper organizer or two. 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, with little organizing principle other than time or program of origin. Why are computers devoid of the organizing principles that allow users to semantically tie together disparate pieces of work on their physical desks?

We may find a measure of control over this chaos by considering the *implicit tasks* informing users' work. A "task" here is a high-level goal towards which a person's actions are directed. Writing an essay, paying bills, or researching camera prices are all examples of tasks. In the physical world, the paraphernalia related to a given active task may be sprawled out across most of a desk, while that of inactive tasks hovers in piles nearby. If the Windows taskbar were meaningfully sorted into tasks, as first explored in Rooms (Jr. & Card, 1986) and later in Greg Smith's GroupBar (Smith et al., 2003), or if task contexts could inform project work, as in UMEA (Kaptelinin, 2003) or TaskTracer (Dragunov et al., 2005)), human spatial memory and hierarchical thinking could be leveraged to help us organize our computational lives.

Although researchers have spent considerable effort on this proposition, the problem is far from solved. Open questions remain: what kind of organizational schemes do users already employ? How can computer users communicate their tasks to the system? Does this need to be communicated explicitly, or can tasks be intuited from user actions? Should specific windows be related to a given task, and how could this be communicated? To what extent can users be troubled to organize such short-lived windows themselves, and to what extent can the computer help organize the users' work?

We have approached these questions in several steps. We began by conducting 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 created a semi-automatic task and window organization system called Taskposé. We describe this system, its motivation and its mechanisms for window relationship tracking. Finally, we evaluate our prototype through an extended user study lasting one week in which participants integrated Taskposé into their normal work practice.

What's In a Window?

It is not immediately obvious what a computer window represents to the user, and why we should care about it at all. First we address the question of "why": here, the answer lies in established practice. The window is an *artifact* – a physical or digital object which users manipulate or reference as part of their workflow. In the digital world, most all artifacts currently being viewed or edited are contained within windows in the operating system. While a wholesale reevaluation of how to organize digital artifacts (Dourish, Edwards, LaMarca, & Salisbury, 1999; Freeman & Gelernter, 1996; Rekimoto, 1999) may yield a considerable gain in workspace understanding and artifact management, we choose to instead take as given the primacy of the window idiom.

Second, complexity arises when considering *what* exactly an open window represents. In our experience observing computer users in situ, we have seen users treat open windows in many different ways, including:

- Actively working on or referencing the artifact it holds
- Keeping the artifact open as a reminder of a to-do
- Representing a process working in the background that is not currently of interest – for example, a virus checker or instant message buddy list
- Keeping a program in computer memory to minimize startup time

We make a simplifying assumption in this work that an open window corresponds to an artifact that is currently either in active use or being referenced. However, we intend to test our system in situations which may include other kinds of windows, so we continue with an eye toward continuing to support diverse work practices while actively improving what we see as the area of greatest potential improvement.

Related Work

Tasks in Cognitive Psychology and Human-Computer Interaction

The idea of task or goal-based analysis of activities is a well-established theory in cognitive psychology. Traditional HCI cognitive modeling, beginning in earnest with *The Psychology of Human-Computer Interaction* (Card, Newell, & Moran, 1983) and reviewed in *Cognitive Architectures and Cognitive Engineering Models in Human-Computer Interaction* (Pirolli, 1999), has generally looked at micro-scale goals and objectives, concerning moment-to-moment activity, whereas computer-based tasks can last hours and involve numerous interrelated tasks. Some psychological work that examines the nature of daily goal/task-based activity includes *Cognitive Psychology of Planning* (Hoc, 1988) and *Plans and the Structure of Behavior* (Miller, Galanter, & Pribram, 1960). Workflow and IT professionals, coordinating with HCI researchers, have likewise explored worker tasks and ad-hoc switching (Bannon, Cypher, Greenspan, & Monty, 1983; Czerwinski, Horvitz, & Wilhite, 2004). Lucy Suchman's *Plans and*

Situated Actions (Suchman, 1987) is cited as another influential work in this area for sounding the call for context (here, the larger task goal) as an important mediator of action.

Defining a *task* has proven a difficult venture, possibly as a result of the fuzzy nature of what really constitutes a computer-based task. The most complete discussion of the theory behind task work in HCI is based on Activity Theory (Bødker, 1991; Kaptilenin & Nardi, 1997; Nardi, 1996), which attempts to place actions in terms of the goals being accomplished and treats the interface as a medium through which this interaction is possible. Winograd and Flores (Winograd & Flores, 1986) would point out, however, that task identification is fundamentally an interpretation forced on an unfocused reality, and that when the user is in a state of *thrownness* ("in the moment"; not reflective) there may be little higher order to users' thinking. This raises the question: how much (or how little) do users think about their tasks when working on a computer? Do they have a full picture of what artifacts are tied to specific tasks, or are their task structures more opportunistic? Could an outside observer, like a computer, pick up systematic queues to answer these kinds of questions automatically, or do users have to constantly and carefully communicate this with the computer?

Two works of note address these questions, and we build on their contribution. Bellotti (Bellotti et al., 2004) was interested in "to-dos" – lists of jobs that needed to be done – and how people go about generating and using them. Most strikingly, she found that 68% of all to-dos created were finished in one week or less, and only a total of 81% were completed by the end of their four-week study. In other words, most tasks were either finished quickly, or not at all; fully one-half of all to-dos not finished within a week were never completed. However, we must be careful not to misappropriate terms here: Bellotti's study concentrated on to-dos rather than "tasks"; to-dos by their nature are reminder-driven rather than action driven, while active tasks are action-driven and rarely reflective.

Hutchings, et al. (Hutchings et al., 2004) set out to study display space usage, but along the way unearthed strong evidence that tasks are composed of multiple windows. According to their study, users spent a median amount of 3.77 seconds on a particular window before switching away; the mean was 20.9 seconds. Of course, this result is mediated by individual programs' window behavior – for example, email clients which place each email in a new window (such as Eudora) will generate drastically more window switches than clients which do not use such popups (e.g. most web mail). ~~However~~Even so, unless users are switching tasks quickly and often, it seems clear that synthesizing multiple sources (as represented by multiple windows) to inform a specific goal is a common operation.

Our work builds on these findings – based on their support, we also assume that users' work can be thought of in terms of tasks.

Tasks and Windows Managers

In the realm of commercial windows managers, there have been literally dozens of incarnations of unique ideas, from open-source XTerm customizations to commercialgrade software, such as the Windows taskbar and Apple OS X Exposé system. Research on task management using these tools is mainly useful here in that it addresses the engineering and design issues upon which to build a window management system (for example: (Bederson & Hollan, 1994; Hutchings et al., 2004)).

Researchers tend to either focus on building systems that can intuit users' task structure, or give up on such automation completely and instead focus on giving users

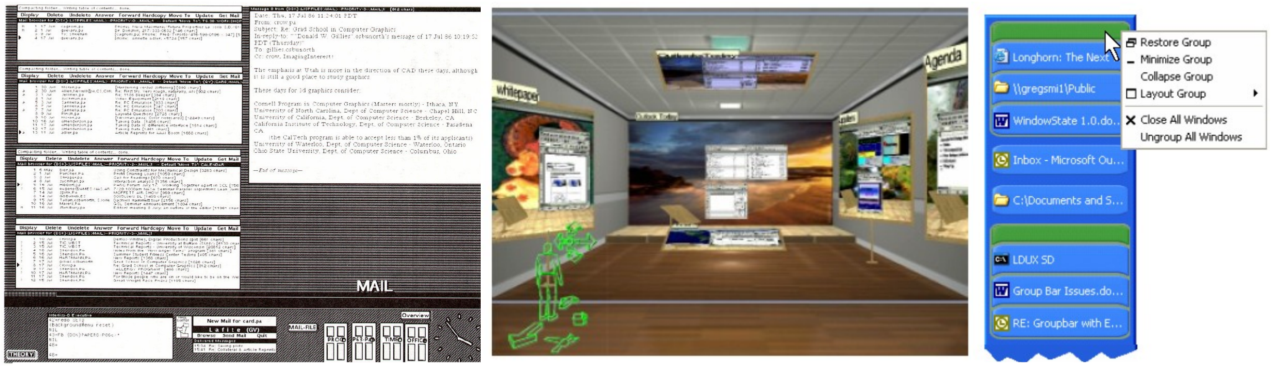
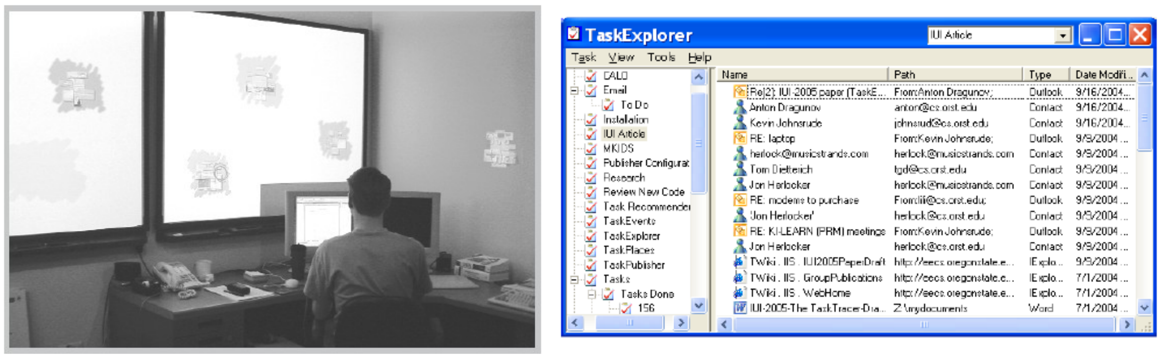


Figure 1. Agnostic window managers: Rooms (left), the Task Gallery (center), and the GroupBar (right).

greater control over manually organizing their workspace. As a result most 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 (Jr. & Card, 1986), virtual desktops, the Task Gallery (Robertson et al., 2000), GroupBar (Smith et al., 2003), the ABC Extension to Windows XP (Bardram, 2005) and Scalable Fabric (Robertson et al., 2004) are all examples of this type of approach (Figure 1). These types of systems offer their most significant return given an equally significant investment in manually organizing tasks. Thus, we believe they are best suited to long-term tasks that operate in a static set of windows. Rather than explore the relative merits of a new kind of agnostic interface, we have chosen to focus on the predictive space.

Predictive window managers make an educated "guess" to automatically 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 they make incorrect decisions, users are generally worse off than if the computer had done nothing at all. At its root, this approach can suffer from a lack of knowledge about the traceable artifacts of task creation and manipulation, as each is forced to choose heuristics based on intuition rather than generalizable theory. Examples of this approach include TaskTracer (Dragunov et al., 2005), Kimura (MacIntyre et al., 2001), SWISH (Oliver, Smith,



Name	Path	Type	Date Modifi...
Re[2]: IUI-2005 paper (TaskE...	From:Anton Dragunov;	Outlook	9/16/2004...
Anton Dragunov	anton@cs.orst.edu	Contact	9/16/2004...
Kevin Johnsrude	johnsrud@cs.orst.edu	Contact	9/16/2004...
RE: laptop	From:Kevin Johnsrude;	Outlook	9/9/2004 ...
herlock@musicstrands.com	herlock@musicstrands.com	Contact	9/9/2004 ...
Tom Dietterich	tgd@cs.orst.edu	Contact	9/9/2004 ...
Jon Herlocker	herlock@musicstrands.com	Contact	9/9/2004 ...
RE: modems to purchase	From:lii@cs.orst.edu;	Outlook	9/9/2004 ...
'Jon Herlocker'	herlock@cs.orst.edu	Contact	9/9/2004 ...
RE: KI-LEARN (PRM) meetings	From:Kevin Johnsrude;	Outlook	9/9/2004 ...
Jon Herlocker	herlock@cs.orst.edu	Contact	9/9/2004 ...
TWiki . IIS . IUI2005PaperDraft	http://eecs.oregonstate.e...	IExplo...	9/9/2004 ...
TWiki . IIS . GroupPublications	http://eecs.oregonstate.e...	IExplo...	7/1/2004 ...
TWiki . IIS . WebHome	http://eecs.oregonstate.e...	IExplo...	7/1/2004 ...
IUI-2005-The TaskTracer-Dra...	Z:\mydocuments	Word	7/1/2004 ...

Figure 2. Predictive window managers: the Kimura system (left) and the TaskTracer interface (right).

Thakkar, & Surendran, 2006) and window-frequency approaches (Nair, Voida, & Mynatt, 2005). UMEA (Kaptelinin, 2003), while not a window manager, follows a similar approach to create dynamically-updating project spaces (Figure 2).

"Fuzzy," Continuously-Changing Tasks

All of the previous work 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 imperfect match for users' mental models.

One interesting result arises from the evaluation of the machine-learning techniques applied to TaskTracer (Stumpf et al., 2005). In their study, the authors asked users to occasionally 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:

> …More interestingly, 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. (Stumpf et al., 2005)

Thus, different contexts will find the same window part of related but conceptually separate tasks. In the TaskTracer example, a document classified under "grant darpa CALO" (the quotes indicate a task title) was later changed to a related "projects CALO" classification when in a different context, and could have changed back if the user returned to working on the DARPA grant (Stumpf, 2005). 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'." This is essentially the problem of asking *pilers* (who, like their namesake, prefer unorganized workspaces) to live in a world where *filing* is the only option (Malone, 1983).

It is important to note here that users should in theory know the classification of every window, since they are aware of their own higher-level goals. But this is clearly *not* the case, as users are having difficulty classifying windows: either the classification systems in use are too sparse to be useful (unlikely because both these systems allow for arbitrary naming of tasks), or the single task classification model does not map well onto users' mental models of their work.

The idea here rings true in our everyday experience as well. 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 simply browsing the web site. From a task perspective, is the user still "buying a book?" Is the user *not* "buying a book?" This situation is certainly a gray area with regards to clean mapping. Or when a second user writes a paper for a conference submission (under a "CHI paper" task), but then later refers to the paper when generating a set of slides for a research group presentation, should the paper still be part of the "CHI paper" task, or not? (Here, we see a situation where an artifact's task association switches due to a context switch.)

Bowker and Star (Bowker & Star, 1999) 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":

> …papers that must be read by yesterday, *but that have been there since last year;* old professional journals that really should be read and even in fact may someday be, *but that have been there since last year;* assorted grant applications, tax forms, various work-related surveys and forms waiting to be filled out for everything from parking spaces to immunizations.

Of course, the phenomenon extends to computers:

> Here the electronic equivalent of "not yet ready to throw out" is also well represented. 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.

The research above supports a hypothesis that the relationship between tasks and actions is not one-to-one and suggests that it is unwise to build a system that strictly categorizes windows into tasks. Our work explores this gray area of task classification. We continue to assume with the rest of the literature that tasks exist, but hope to show that adding a bit of complexity to the task model will translate to task-based systems which map better to users' mental models of their work.

Open Questions

There are important differences in task structure assumed by the task classification approach and by our less boundary-oriented approach. Beginning with the classification approach, we may use a tree structure to characterize tasks by representing the dependencies that typically appear within and between tasks as viewed from a user's perspective. One such task, for writing this paper, might look as in Figure 3:

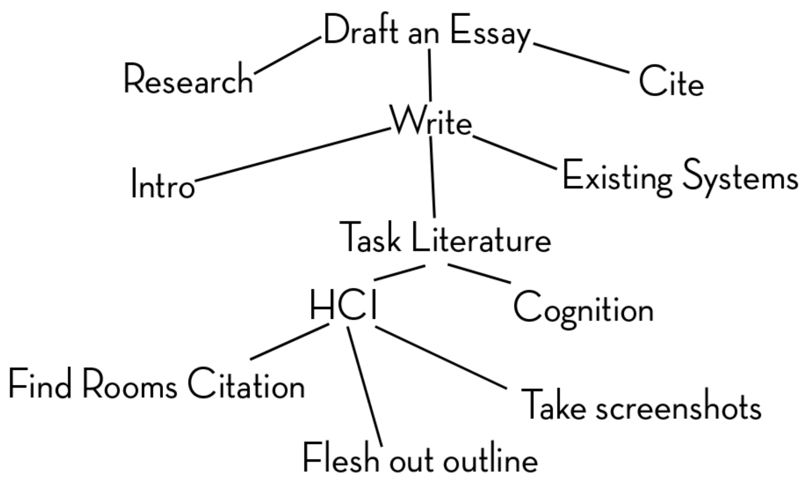


Figure 3: An example task tree

Here we see a task in progress. There is a complex hierarchy of information associated even with the seemingly-straightforward task above. The tree is expanded along one path down to one level before we might expect to see individual windows appear. For instance, the "Find Rooms Citation" task might include an Internet browser window open to Google Scholar, an Adobe Acrobat window with the paper itself, and perhaps another Internet window open to a style guide web site.

This kind of structure cannot be used once tasks interdependencies are introduced. For example, how do we represent the fact that finding the Rooms citation might also be important to writing the Cognition literature section of the paper? A directed graph (digraph) may be a better representation. For example, research on cognition might also require that the writer look up the Rooms citation, as in Figure 4:

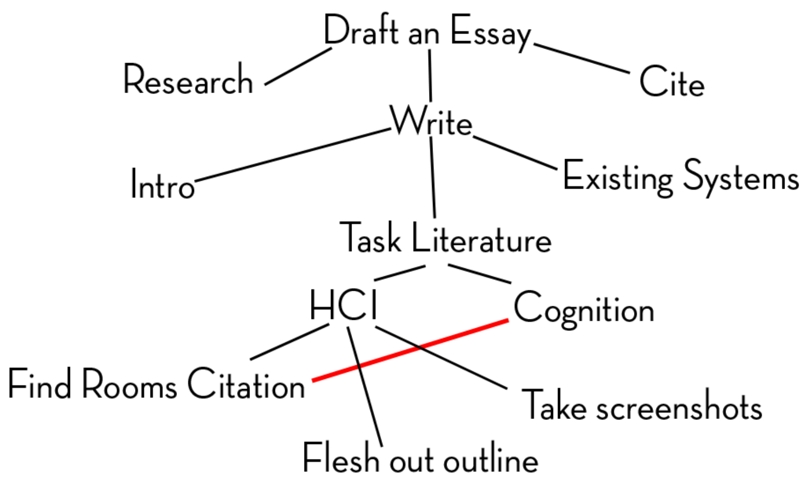


Figure 4. The task tree, with additional complexity requiring a digraph

By opening the door to such complexity, we find no easy way to contain it. For example, what if the HCI Outline and the Cognition section were dependent on each other, as might happen if information gained in researching the Cognition section helped inform the HCI outline but we also wish to frame the Cognition section itself by the categories described in the HCI outline. Is a cyclic graph appropriate here? At what stage do we simply give up the ghost?

We pause to note that several dimensions of task classification are exemplified in the foregoing discussion, summarized by these questions that could be asked about any user's tasks:

- Are the tasks connected to the graph in a strict fashion (to one parent), or do they cross-pollinate and have multiple parents?
- Are the "leaf nodes" (the windows) only a few levels down, or are they deeply nested in a series of sub-tasks?
- Are particular tasks *persistent* or *transient*? *Persistent* tasks either repeat or take place over relatively long periods: for instance, "feed the dog." *Transient* tasks happen once and then disappear, such as "return Professor Jones's email."
- Are the tasks' compositions *discrete* or *continuous*? A *discrete* task has a set of steps necessary for completion ("make dinner"), whereas a *continuous* task is not so structured ("check up on advisees' progress on the project").
- To what extent are tasks interrupt-driven, and to what extent are they opportunistic?
- Do users often interrupt their current tasks to begin a new one because they find that they need new information?
- How much of a "task tree" does a user keep in working memory at any time? The entirety or only those elements surrounding the current focus?

As is often the case with attempts to classify human cognition, a strict representation such as the one above will often break down in the face of complex "real" cognition. However, because these kinds of questions are new to the literature, it is as yet unclear which parts of the theory require significant adjustment, and which only require that a little "fuzz" be added. A serious study of users' approach to tasks on a computer will inform us a great deal in this regard.

Fieldwork

Goals

The goal of this exploratory study was to produce a model for task creation and manipulation. We were particularly interested in patterns that carry across interaction styles, such as situations that prompt users to create new subtasks, as well as existing coping mechanisms for window management. We hoped to capture user mistakes and breakdowns, as they are often a useful starting place for future designs. We observed whether users were focusing on a single task with contextual tasks in the background; whether their "task trees" were shallow or deep; whether there were useful task artifacts left as a result of these users' interactions, and so on. By pulling out cross-user threads from this experiment we hoped to be able to provide evidence to guide our research.

Method

The study was a simple in-situ observational visitation. We recruited subjects who came to use a public computer cluster at Stanford University. Participants were of both genders, and mainly consisted of undergraduate and graduate students. We asked their permission to enter our subject pool. Then, at some point (arbitrarily chosen) during their ongoing work, we observed and video-taped their actions, occasionally asking them questions about their ongoing activity. Users with multiple windows open were preferentially chosen for this interruption. Each of the 19 participants was observed and videotaped for a 5-15 minute period of normal computer use, with occasional interruption from the researcher for an explanation of the user's idea of his or her workspace.

Results

Even though we preferentially chose subjects with multiple open windows, most of the participants exhibited limited evidence of multitasking. We believe this may be due to contextual variables, especially the fact that the experiment was conducted in a public computer cluster rather than on users' home machines. Public computers are generally used for short periods of time and for single purposes such as checking email; this environment discouraged multitasking. Few participants remained on the public computers for more than an hour. Additionally, these computers are rarely outfitted with the specialized programs that many users leave open while multitasking, such as e-mail and IM chat clients. 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.

In general, among the observed multitasking behaviors, users organized their work along task lines. Multitasking often involved one central task (possibly spanning several windows) and a series of peripheral tasks. In Figure 5, a participant writes an email to her friends about a concert by referencing a web site:

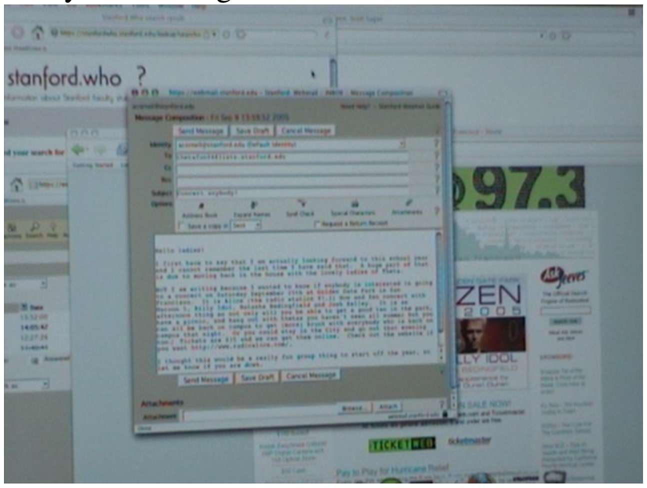


Figure 5. A participant writing an email (foreground) about a concert (background bottom right), with additional other windows open in the background.

Task switching is often opportunistic. In Figure 6, a participant editing his resume switches to a chat window when his friend greets him, then to several other unrelated chat windows before returning to work. Chat windows are emblematic of a class of window 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 in certain circumstances email clients (especially those with popup notification).

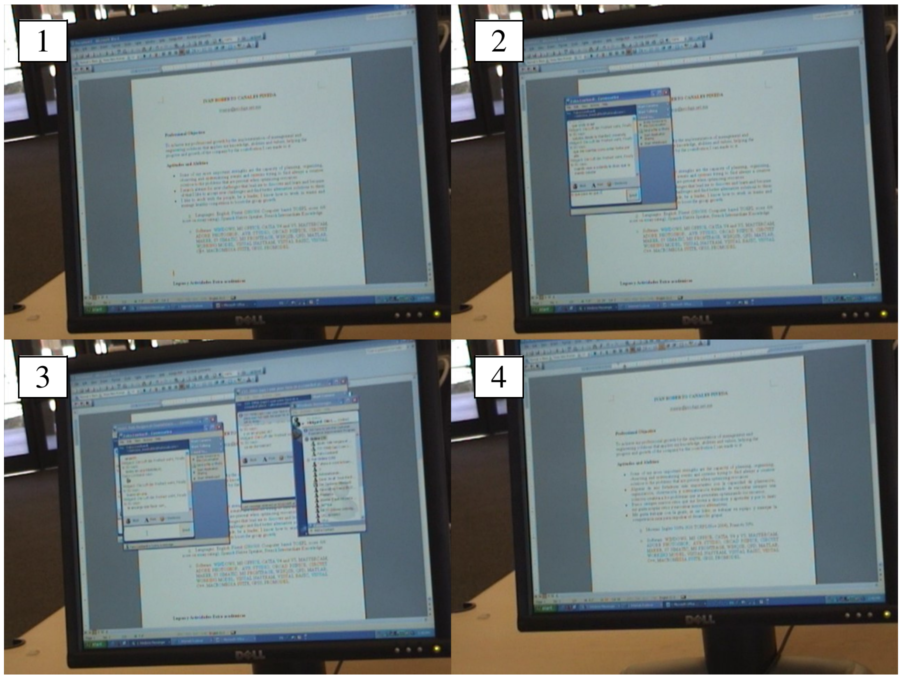


Figure 6. A study participant is distracted by a series of messaging windows, and then returns to work.

Users trying to consolidate large bodies of information into a meaningful whole usually had the most windows open. In Figure 7, a participant using Firefox tabbed browsing holds six tabs open in addition to a Microsoft Word document he is pasting text into. (Tabs are another interesting special case as a window management technique – they collapse many artifacts, often unrelated, into a single window.)

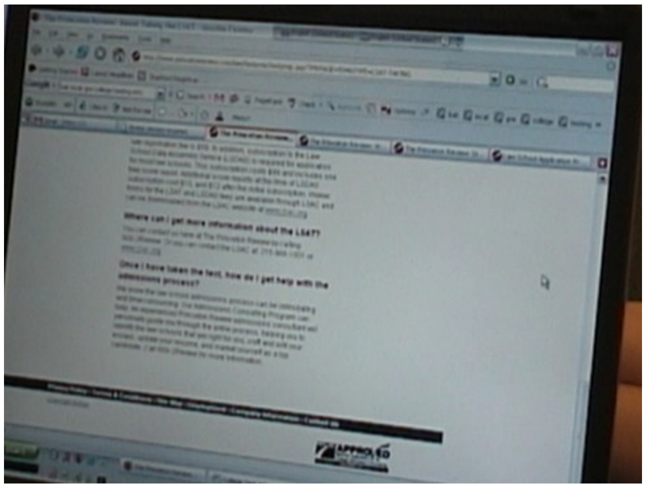


Figure 7. There are six Firefox tabs and several programs open for this user.

Using the recorded video tape, we analyzed the most interesting video sessions for activities such as window switching and dwell time. We have used this data as a test set of data for Taskposé. Appendix I contains an example coded data set.

Discussion of the Observational Study

Rather than holding entire ‘task graphs’ in working memory, users generally are aware of a single task at any given time and whatever is currently the focus of attention takes priority over all else. As a result, ‘orphaned’ windows are often left open long after they are still in use, because users tend to forget about them and they do not make themselves apparent. The task graph should therefore be considered ‘random access’, or perhaps ‘center-surround’ (Furnas, 1986) where the user is aware of what the current task, and only the sub- or super-tasks that are especially relevant to the project at hand.

A cognitive 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 threads such as chat, music or email are not uncommon.
- New tasks or subtasks are spun off opportunistically, as needed. 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

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 of gray, which strict groupings cannot support, and
2. Strengths of association between artifacts may change (slowly) over usage time, or immediately if the context switches.

Other research into task-based windows managers, both agnostic and predictive, has tended to assume that windows are always *cleanly mapped* into a specific task and that windows are *statically* part of one task. For example, the GroupBar allows users to place windows into one group, and does not provide support for multiple simultaneous classifications. Similar claims could be made about the Task Gallery, Scalable Fabric, Rooms, and others.

Following computer science terminology as well as Bowker and Star (Bowker & Star, 1999), 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.

As existing research into task-oriented windows managers has been classificatory, we can build on previous work by exploring the design of associative window managers. Our goal, then, is to design a proof-of-concept window manager which incorporates association in a useful and user-friendly way.

Taskposé

Design Goals

As we have seen, human-computer task activities are characterized by inherent complexity. We believe that this complexity can best be understood from the standpoint of the following *associative,* and not *classificatory,* assertions:

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

In order to support complex desktop activity, we have developed an associative window manager wherein window icons appear in a two-dimensional space. An outline of its workings appears in Figure 8.

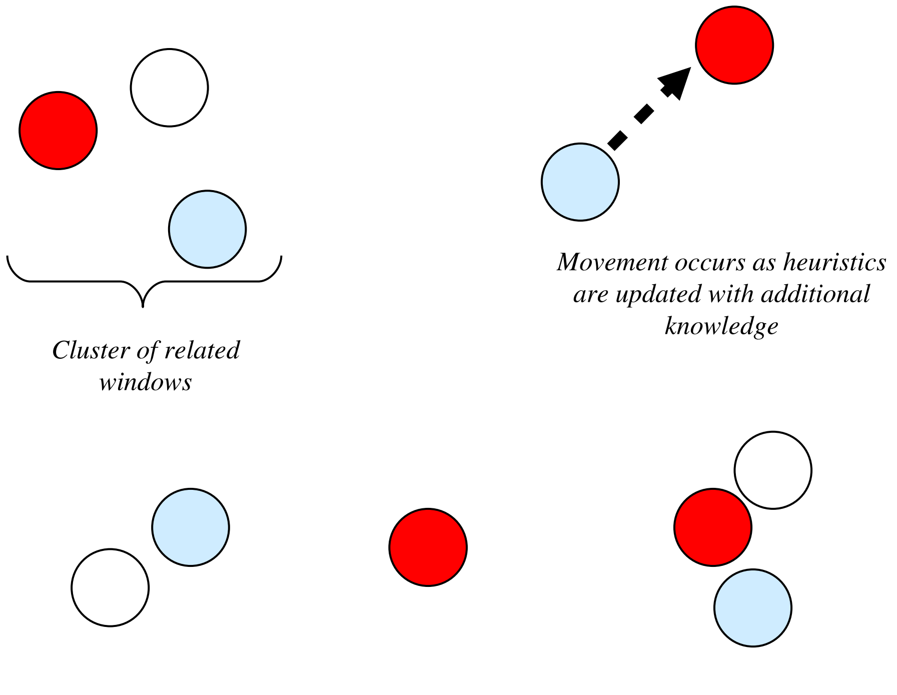

Cluster of related windows
Movement occurs as heuristics are updated with additional knowledge

Figure 8. Sketch of the system mechanics

The goal of Taskposé is to visually cluster related windows so that users can manage their complex computer activities. It represents open windows by icons where the distance between these icons is tied to the predicted "semantic" (i.e., task-based) relation between the windows represented by the icons. 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 windows may participate in multiple tasks. For example, a window related to writing a paper and to paying bills 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 live "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 windows titles arranged in a one-dimensional line such as in the Windows taskbar. However, we acknowledge that it is an open question whether two dimensions, one, or neither is most natural for associative window managers. Additionally, Taskposé may eventually live as a stand-alone application on a small separate monitor, as multiple-monitor systems become more and more prevalent. Given the prevalence of single-monitor users today, we chose to instead apply this approach to fashion a full-screen view like Apple's Exposé (the system's namesake).

Interaction with Taskposé

The Taskposé user interface is shown in Figure 9. Each open window is represented by a screenshot, a program icon and a window title. Windows are not classified into tasks or activities, but over time move near each other if Taskposé believes them to be related. Because no classifications are made, ambiguity and context-dependence can be preserved when necessary.

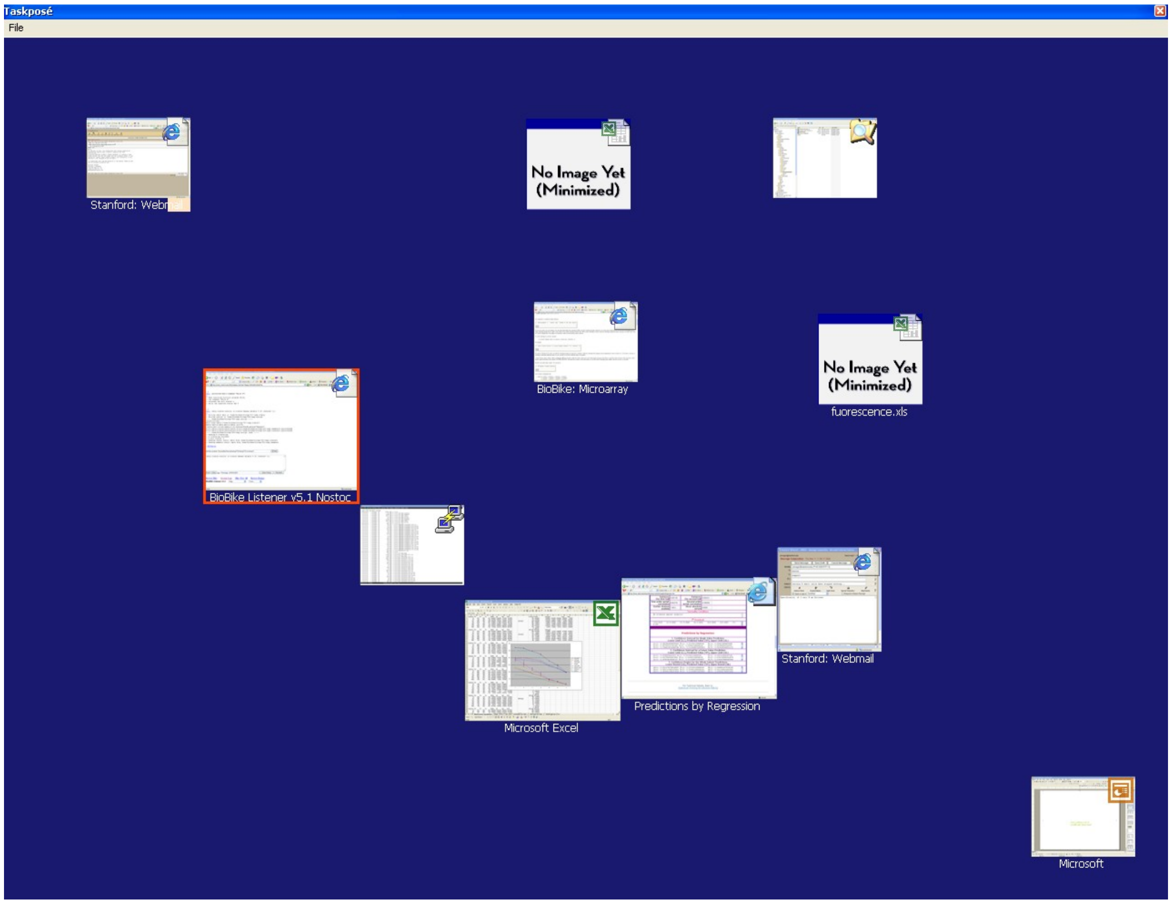


Figure 9. The Taskposé Visualization

The Taskposé Visualization

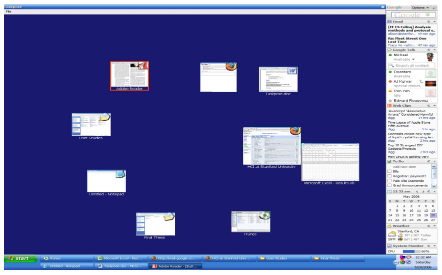


Figure 10. Two strongly related windows appear close to each other in the visualization.

A few simple rules guide Taskposé's user interface. *Related* windows appear closer together the more related Taskposé believes them to be. So, tightly bound windows will move right next to each other (Figure 10), and unrelated ones force each other far 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. The user can move a window to another part of the visualization via a drag interaction if he or she wishes. However, Taskposé does not currently interpret drag-and-drop location as new relationship information. A user may "anchor" a window via a right-click interaction, preventing it from moving until unanchored. This small piece of customization was intended to allow users who wished to keep tasks in specific areas of the visualization to do so.

Important windows inform the 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. That is, Taskposé displays important windows larger than normal, and shrinks unimportant windows.

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 which users will want to find, and will be the most disruptive if they unexpectedly shift. Unimportant windows were assumed to be less disruptive to move. The user does not currently have any method to override Taskposé's importance beliefs.

Switching Windows

A complete interaction with Taskposé takes only a few seconds. However, because of the variety of interaction styles that users incorporate, there are several methods of working with the system. The Taskposé visualization may be brought up in one of two ways:

1. Double-clicking the Taskposé icon in the System Tray. This interaction is intended to be used by the visual user, or in setting where the user is already at the mouse.
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 – the ` key is placed directly above Tab on most American keyboards. Alt-` is the faster expert key combination, and especially so when the left hand is on the keyboard.

When called, Taskposé covers the entire screen with its visualization, similarly to Apple's Expose. The current window is outlined in red (Figure 11), to help the user orient him or herself in the visualization. Taskposé takes continuously updating screenshots of its windows while the user is choosing. By clicking on a window, the user signals that he or she wishes to switch to that window. Taskposé hides and immediately switches to the window. If the user decides not to switch windows, he or she can hide the visualization by repeating either of the show mechanisms above.

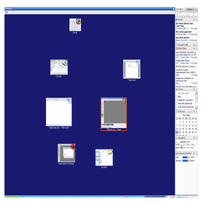


Figure 11. The most recent window of attention is surrounded by a red border to facilitate visual search.

Implementation and Algorithms

The Taskposé prototype is implemented in C# under the .NET platform, and hooks into the Win32 API to listen to and publish window events, as well as retrieve window icons, labels and screenshots. Three main algorithms underlie the Taskposé system: the WindowRank algorithm for determining window importance, the window relationship heuristic, and the graph layout algorithm. Our goal in creating these algorithms was to fashion a "proof of concept" system because the existing algorithms output binary decisions rather than continuous values. The algorithms' success will be evaluated as part of Taskposé's user studies.

The WindowRank Algorithm: Determining Window Importance

The WindowRank algorithm is responsible for determining window importance. It 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. Other algorithms have attempted to utilize window switching to determine window relevance with reasonable success (Nair et al., 2005; Oliver et al., 2006), but to our knowledge none have attempted to do so to describe window *importance*. We later use this importance metric to inform our relatedness heuristic.

WindowRank is best described by analogy to Google's PageRank (Page, Brin, Motwani, & Winograd, 1998), its intellectual inspiration. 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. A page then adds its own PageRank to every page it links to by taking its summed PageRank, dividing by the number of outgoing links and adding that amount to each linked page's PageRank. This algorithm is run iteratively until the entire system stabilizes.

WindowRank treats windows as nodes in the graph and user window switches as edges, analogously to PageRank's web pages and links. 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. Pseudocode for the WindowRank algorithm follows:

```
for(int iteration=0; i<ITERATIONS; i++)
{
      Number incomingRank = 0;
      Window winA = next window;
      foreach(Window winB that winA has switched to)
      {
            incomingRank += winB.WindowRank * (Switches from winB
                                 to winA / Total Switches from winB
                                 to any window);
      }

      winA.WindowRank = (1 – BUFFER) + (BUFFER * incomingRank);
}
```

In practice, we have found a value of 40 for `ITERATIONS` to be more than sufficient to stabilize the WindowRanks of windows, especially given that this algorithm is being run on already-stable values from the previous update cycle and is given only a small delta of an extra switch or program event. We follow Page *et al.* in using a `BUFFER` constant of .85 in our algorithm.

There are a few major differences between PageRank and WindowRank. First, PageRank prunes "dangling nodes" – pages without outgoing links – until the very end of the algorithm because they will accumulate PageRank from their incoming links but not disperse it to any outgoing links, unbalancing the calculation. This tactic is appropriate for PageRank, which runs on an unchanging set of web pages, but given the dynamic nature of new windows opening and closing, WindowRank would be almost useless if every new window had to be removed from the algorithm. Taskposé cannot assume that there have been any "links" created when it opens. To avoid this potential pitfall, we assume that each "dangling" window has a roughly equal chance of switching to any other window, and thus start each new window with one vote for every other window open. The net result of this is that new or dangling windows simply redistribute their WindowRank equally amongst the rest of the windows.

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 is quietly run every time a new user action (switch, open, close, etc.) occurs. Because the number of graph nodes is relatively small, the algorithm in practice runs quite quickly, and is not a performance concern. Second, 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 (see the section on the window relationship heuristic for further discussion).

The Window Relationship Heuristic: WindowRank Applied

The most important utilization of WindowRank appears in the window relationship heuristic. It was our goal to fashion an algorithm which would output relatedness (association) over a continuous region, which we could then incorporate into our visualization. Thus, our heuristic 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. This heuristic is similar to other window switch relationship heuristics (e.g. (Nair et al., 2005; Oliver et al., 2006)), but is unique in its incorporation of window importance. We believe that this consideration is important to improving the accuracy of such algorithms.

WindowRank is necessary here because Window A and Window B may have different opinions about how closely related they are to each other. For an illustration, consider a naïve heuristic which treats both Window A and Window B as equals in the decision. It would likely do something like:

```
Number AVote = Switches from A to B / Switches from A to any
window;
Number BVote = Switches from B to A / Switches from B to any
window;
return (AVote * 0.5) + (BVote * 0.5);
```

This approach breaks down in some circumstances. If Window A is an important window (with high WindowRank), it will likely have switched to and from several different windows many times. So, its `AVote` will be relatively small, but also probably 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 (has been switched to only once or twice), and then switches to Window A, `BVote` is going to be extremely high because `Switches from B to any window` will be small. Thus, by averaging `AVote` and `BVote`, the 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 heuristic in use weights each vote by the ratio of its rank to the two windows' ranks summed:

```
Number AVote = Switches from A to B / Switches from A to any
window;
Number BVote = Switches from B to A / Switches from B to any
window;
Number ARatio = winA.WindowRank / (winA.WindowRank +
winB.WindowRank);
return (AVote * ARatio) + (BVote * (1 – ARatio));
```

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 algorithm lays out the icons. We decided upon a spring-embedded graph because of its aesthetic layout characteristics and its ability to map continuous values from our relatedness heuristic directly onto visual distances. It accomplishes this mapping by bringing nodes connected by short springs close together, and forcing nodes connected by long springs to be farther apart.

In a spring-embedded graph layout, a simulated spring is attached between every two nodes in the graph. Each of these springs has an associated natural length and spring constant k, which represents the resilience of the spring. These two properties can be manipulated by the system to generate desired clustering qualities – for example, two windows connected by a short, stiff spring will stay near each other. Each spring's force acts on its two endpoints *A* and *B* according to Hooke's Law:

$$F_{V_A V_B} = -k|x_{V_A} - x_{V_B}|$$

Then, the total force on a vertex can be calculated by summing the forces from all n outgoing springs:

$$\texttt{Total Force on Vertex A} = \sum_{x=1}^{n} F_{V_A V_x}$$

During each program cycle, the window is moved by an amount proportional to the overall force acting on it. This proportion is determined by each window's WindowRank. That is, windows with above average WindowRank (important windows) move less, and windows with less WindowRank (unimportant windows) move proportionately more. This allows the graph to smoothly update without disrupting the positions of important windows. In Figure 12, we see the graph with edges displayed. The numbers represent the relative forces on the springs:

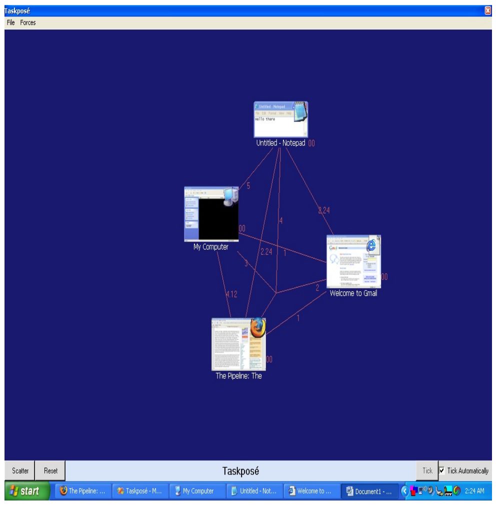


Figure 12. Graph with edges displayed.

The program clock ticks at a default rate of one tick every 150 milliseconds. At each tick, all window icons move as determined by the sum of the spring forces. Thus, over a short period of time, an initially disorganized graph will migrate into an organized pattern as a side effect of trying to minimize the total energy of the system, guided by the heuristic analysis of the windows' descriptions and user actions. As the heuristics change the associations between the windows, the window icons in the visualization adjust to reflect the change.

The window relationship heuristic does a simple update to achieve its desired result. First, it computes the mean and standard deviation of these window relationship values across the graph, as calculated by the window relationship heuristic. Then, for each pair of windows, it scales the spring stiffness and length based on how many standard deviations the relationship value is from the computed mean. It attempts to create a "normally distributed" (term used loosely) set of spring lengths and stiffnesses:

```
/* Not shown: compute mean and stdDev */
Number unitsFromMean = (RelationshipHeuristic(WinA, WinB) - mean)
                       / stdDev;
Number strength = .5 + SPREAD*unitsFromMean;

// Make sure this stays between 0 and 1.
Number boundedStrength = Maximum(Minimum(strength, 1), 0);

// Now make the spring shorter the closer they're
// related, and make the spring stiffer.
Edge = Graph edge between WinA and WinB;
Edge.setSpringLength(DEFAULT_SPRING_LENGTH * (1 -
                       boundedStrength));
Edge.setSpringK(DEFAULT_SPRING_K * boundedStrength);
```

The result of this operation is that closely related windows end up with short, stiff springs, and unrelated windows end up with long but loose springs. The looseness is desired for unrelated windows so that there is some flexibility in the windows' relative placement.

Evaluation

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 allow users to give unbiased evaluations of H1 and H2.

We conducted two studies of Taskposé, encompassing two separate study designs (a first-use study and a longitudinal study) and four different types of data collection (free form interview, self-reported questionnaire, videotaped observation and computer-generated usage log).

First-Use Study

We chose a first-use study for its power to rapidly elicit usability problems. Ten undergraduate students at Stanford 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.

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 the following information about the Political Science programs at four major universities:

- Department Address
- National Departmental Ranking
- Three professors' names, biographical information, and publications

This information was to be compiled into a separate Microsoft Word document for each Political Science program. Participants were given 20 minutes to complete the task. At the end of the task, participants were paid $10 for their assistance.

The task required numerous window switches and caused a great deal of window thrashing (Jr. & Card, 1986). 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 the wild" by users over an extended period would produce a compelling measure of its success or failure. The main strength of a longitudinal approach lies in testing the sustainability and scalability of our platform; its main drawback is that allowing users to use the software on their own time precludes a researcher from observing the interaction.

Ten undergraduate students (5 male, 5 female) were recruited for this study. Taskposé was installed on their main computers, and the researcher demonstrated its use. Participants were asked to use Taskposé in the course of their everyday computer work for an hour a day, over seven days. (Participants who wished were allowed to use Taskposé more than the required seven hours.) We paid participants \$62.50 for their weeklong participation.

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. 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.

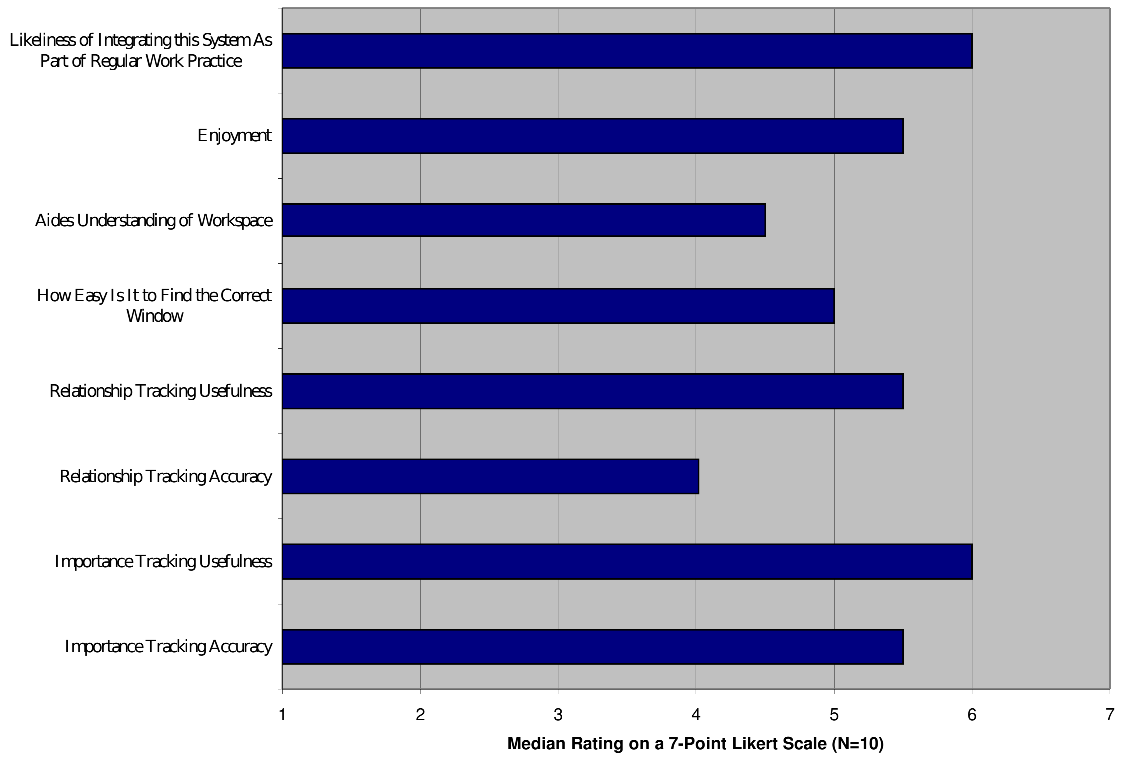


Figure 13. Self-reported user ratings of the Taskposé system. Generally, users reacted more positively to the system design than to the implementation of the importance and relationship algorithms.

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 13). Enjoyment was likewise rated highly.

Users' willingness to integrate an idealized system into their regular workflow, especially given the limits they had experienced with the window relationship heuristics, 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, suggesting 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 (when alt-tab was preferred).

Window Importance and Relationships

Taskposé's algorithms were found to be powerful enough to allow evaluation of the rest of the program, though they were limited in their accuracy and in fact the most common target of system criticism. On a 7 point Likert scale, users rated the importance tracking algorithm (WindowRank) a median 5.5 and the window relationship heuristic a median 4.0.

Freeform comments revealed a few classes of problems with the relationship heuristic. First, "parent program" relationships were not accounted for in the Taskposé model: AIM chat windows would thus not automatically group with each other and with the buddy list, but instead place themselves randomly in the visualization. While our analysis of task behavior in the fieldwork study indicated such a parent-child relationship would not always be desired (consider one chat related to a term paper and another related to a book purchase), its availability as a default seemed preferred. Secondly, participants reported that when working on multiple tasks, they found Taskposé would cluster all 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, tabbed Internet browsing was found to decrease the usefulness of the heuristic, as such an Internet browser started associating with multiple groupings.

Logging Results

Eight of the ten participants returned usable activity logs (one user accidentally deleted his logs, and another's logs were corrupted). We attempted to analyze this log data to gather more quantitative usage information. Specifically, we had intended to measure how often participants switched programs using Taskposé, and compare that to the number of times they switched using other means (taskbar, alt-tab, or simply clicking on the window). Surprisingly, because users left the Taskposé software running in the background for extended periods of time (one user logged 195 hours) and sometimes used it continuously, 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 remember window relationships when it is turned off, so every time the software was closed it took a few minutes before Taskposé starting grouping windows intelligently. Though we are left with artificially low usage rates as a result of this behavior (Table 1),

User	*Total Number of Window Switches Using Taskposé*	*Total Time Running Taskposé (hours)*	*Taskposé Switches per Hour Using Taskposé*
1	N/A	N/A	N/A
2	19	12.10472	1.569635
3	196	195.4275	1.002929
4	181	57.93083	3.124416
[illegible]	[illegible]	[illegible]	[illegible]
9	48	40.75222	1.17785
10	N/A	N/A	N/A

Table 1. Usage log data analysis, highlighting the unexpectedly high number of hours users spent with Taskposé running. Required time was 7 hours.

it is an encouraging suggestion as 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 improvements based on the long-term evaluation.

Users tended to either use the anchoring functionality extensively or not 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 (Figure 14) to make the interaction more visible (Hoeber, 2005). In our envisioned prototype, a specific corner of each window would be treated as a "hotspot" to allow easier (one-click) anchoring at the current location.

Figure 14. Prototype of a more visible "pin" interaction for anchoring.

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 head.

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 heuristic.

Conclusions and Future 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 almost universally only able to make classification decisions rather than generate the real-valued strength-of-association numbers that Taskposé requires, so significant rethinking 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 (Cox & Cox, 2000) could also prove fruitful.

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 difficult problem and a direction for future research.

To conclude, we reconsider the big picture question: how to organize our complex world. As the digital re-fuses with the physical (Ishii & Ullmer, 1997; Weiser, 1999), the *artifacts* of attention will cease to be pixel-bound windows and begin to encompass physical and embodied interactions. How do we support interaction in this mixed world? Papers and books automatically shuffling around and reorienting themselves on a physical desk would be considerably disorienting, but the main contribution of this thesis, the adoption of an associative rather than classificatory approach to interactive task work, would still be applicable. It might even lend new meaning to the concept of having data literally "at our fingertips."

Acknowledgements

This particular task was an undertaking spanning more than a year of independent work with Jeff Shrager. He has at various times driven out to meet me on an off day, called me from home, lent me his own personal recording equipment and installed my software on his computer to videotape himself trying it for me. I am deeply indebted to Jeff for his wisdom, humor, and quick email response time throughout the process.

Intellectually, this paper has benefited from discussions with a great number of professional HCI researchers. I'd like to thank Terry Winograd, Scott Klemmer, Todd Davies, Erica Robles, Victoria Bellotti, Jim Thornton, Ed Chi, Duke Hutchings, Simone Stumpf, Jon Herlocker and the TaskTracer team, Greg Smith, Mary Czerwinski, Pat Langley, Stephen Voida, Jim Hollan, Adam Perer and Bjoern Hartmann for their time and energy. Though Taskposé was programmed by the author, in earlier incarnations it was supported by code from Jon Herlocker (TaskTracer) and Duke Hutchings. Many thanks to them for their help. Finally, a blanket thank you to the Stanford HCI Group for teaching me how to think like an HCI researcher.

Emotional support is necessary for any undertaking of this magnitude. To my peers and teachers in the Symbolic Systems program at Stanford: thank you for the community. To my parents, Neila and Andy, and to my girlfriend, Adi: thank you for everything.

Work on the Taskposé project has been supported by a URP Minor Grant. The studies in this paper have been authorized by Stanford's IRB board, protocols 2734 and 4804.

References

Bannon, L., Cypher, A., Greenspan, S., & Monty, M. L. (1983). Evaluation and analysis of users' activity organization. *CHI '83: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Boston, Massachusetts, United States. 54-57.

Bardram, E. (2005). Activity-based computing: Support for mobility and collaboration in ubiquitous computing. *Personal Ubiquitous Computing, 9*(5), 312-322.

Bederson, B. B., & Hollan, J. D. (1994). Pad++: A zooming graphical interface for exploring alternate interface physics. *UIST '94: Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology,* Marina del Rey, California, United States. 17-26.

Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G., & Ducheneaut, N. (2004). What a to-do: Studies of task management towards the design of a personal task list manager. *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Vienna, Austria. 735-742.

Bødker, S. (1991). *Through the interface: A human activity approach to user interface design*. Hillsdale, N.J.: L. Erlbaum.

Bowker, G. C., & Star, S. L. (1999). *Sorting things out: Classification and its consequences.* MIT Press.

Card, S. K., Newell, A., & Moran, T. P. (1983). *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, Inc.

Cox, T. F., & Cox, M. A. (2000). *Multidimensional scaling.* Chapman & Hall/CRC.

Czerwinski, M., Horvitz, E., & Wilhite, S. (2004). A diary study of task switching and interruptions. *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Vienna, Austria. 175-182.

Dourish, P., Edwards, W. K., LaMarca, A., & Salisbury, M. (1999). Presto: An experimental architecture for fluid interactive document spaces. *ACM Trans.Comput.-Hum.Interact., 6*(2), 133-161.

Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. L. (2005). TaskTracer: A desktop environment to support multi-tasking knowledge workers. *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces,* San Diego, California, USA. 75-82.

Freeman, E., & Gelernter, D. (1996). Lifestreams: A storage model for personal data. *SIGMOD Rec., 25*(1), 80-86.

Furnas, G. W. (1986). Generalized fisheye views. *CHI '86: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Boston, Massachusetts, United States. 16-23.

Hoc, J. -. (1988). *Cognitive psychology of planning.* Academic Press Professional, Inc.

Hoeber, Tony. (2005). *Face to face with Open Look.* Retrieved June 12, 2006, from http://www.guidebookgallery.org/articles/facetofacewithopenlook.

Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., & Robertson, G. (2004). Display space usage and window management operation comparisons between single monitor and multiple monitor users. *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces,* Gallipoli, Italy. 32-39.

Ishii, H., & Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* , 234-241.

Jr., D. A. H., & Card, S. (1986). Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Trans.Graph., 5*(3), 211-243.

Kaptelinin, V. (2003). UMEA: Translating interaction histories into project contexts. *CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Ft. Lauderdale, Florida, USA. 353-360.

Kaptilenin, V., & Nardi, B. A. (1997). *Activity theory: Basic concepts and applications.* Retrieved 01/09, 2006 from http://www.sigchi.org/chi97/proceedings/tutorial/bn.htm

MacIntyre, B., Mynatt, E. D., Voida, S., Hansen, K. M., Tullio, J., & Corso, G. M. (2001). Support for multitasking and background awareness using interactive peripheral displays. *UIST '01: Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology,* Orlando, Florida. 41-50.

Malone, T. W. (1983). How do people organize their desks?: Implications for the design of office information systems. *ACM Transactions on Information Systems (TOIS), 1*(1), 99-112.

Miller, G., Galanter, E., & Pribram, K. (1960). *Plans and the structure of behavior.* Holt, Reinhart & Winston Inc.

Nair, R., Voida, S., & Mynatt, E. D. (2005). Frequency-based detection of task switches. Paper presented at the *Proceedings of the 19th British HCI Group Annual Confernece,* Edinburgh, Scotland. pp. 94.

Nardi, B. A. (1996). *Context and consciousness: Activity theory and human-computer interaction.* MIT Press.

Oliver, N., Smith, G., Thakkar, C., & Surendran, A. C. (2006). SWISH: Semantic analysis of window titles and switching history. *Proceedings of the 11th International Conference on Intelligent User Interfaces*, 194-201.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). *The PageRank Citation Ranking: Bringing Order to the Web,* Pirolli, P. (1999). Cognitive engineering models and cognitive architectures in human-computer interaction. In F.T. Durso, et al. (Ed.), *Handbook of applied cognition* (pp. 441). West Sussex, England: John Wiley & Sons.

Rekimoto, J. (1999). Time-machine computing: A time-centric approach for the information environment. *UIST '99: Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology,* Asheville, North Carolina, United States. 45-54.

Robertson, G., Dantzich, M. v., Robbins, D., Czerwinski, M., Hinckley, K., & Risden, K. et al. (2000). The task gallery: A 3D window manager. *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* The Hague, The Netherlands. 494-501.

Robertson, G., Horvitz, E., Czerwinski, M., Baudisch, P., Hutchings, D. R., & Meyers, B. et al. (2004). Scalable fabric: Flexible task management. *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces,* Gallipoli, Italy. 85-89.

Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., & Robbins, D. et al. (2003). GroupBar: The TaskBar evolved. *Proceedings of OZCHI 2003,*

Stumpf, S. (2005). *Email Discussing TaskTracer Results*

Stumpf, S., Bao, X., Dragunov, A., Dietterich, T., Herlocker, J., & Johnsrude, K. et al. (2005). Predicting user tasks: I know what you're doing! *Twentieth National Conference on Artificial Intelligence (AAAI-05),*

Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication.* Cambridge University Press.

Weiser, M. (1999). The computer for the 21st Century. *ACM SIGMOBILE Mobile Computing and Communications Review, 3*(3), 3-11.

Winograd, T., & Flores, F. (1986). *Understanding computers and cognition: A new foundation for design.* Ablex Publishing Corporation.

Appendix I: Example Coded Video Data Set

Participant	18
Video	2
Time In	1:09:13
Time Out	1:13:13
Total	0:04:00
Characterization	Compiling internet documents into a MSWord

Label	Program	Task	Notes
A	IE	Tests	Looking for information on LSAT, GRE, etc.
B	MSWord	Tests	Compiled document
C	IE	Tests	Second window w/ information on tests
D	IE	Tests	Third window w/ information on tests
E	IE	Tests	Fourth window w/ information on tests
F	IE	Mail	Gmail
G	IE	Axess	Axess
H	IE	News	CNN.com
I	IE	Mail	Webmail
J	Desktop App		Has a magic hat

Switch To	Time at switched away	Time Spent on Window (s)	
A	1.09.23	10	
B	1.10.04	41	
A	1.10.06	2	closed A
C	1.10.10	4	closed C
D	1.10.17	7	closed D
E	1.10.58	41	
F	1.11.08	10	
G	1.11.12	4	closed G
E	1.11.15	3	