User Interface Models for the Cloud

Hubert Pham
MIT CSAIL
32 Vassar Street
Cambridge, MA 02139 USA
hubert@mit.edu

ABSTRACT
The current desktop metaphor is unsuitable for the coming age of cloud-based applications. The desktop was developed in an era that was focused on local resources, and consequently its gestures, semantics, and security model reflect heavy reliance on hierarchy and physical locations. This paper proposes a new user interface model that accounts for cloud applications, incorporating representations of people and new gestures for sharing and access, while minimizing the prominence of location. The model's key feature is a lightweight mechanism to group objects for resource organization, sharing, and access control, towards the goal of providing simple semantics for a wide range of tasks, while also achieving security *through* greater usability.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors, Security

Keywords: Cloud, Desktop, Groups, Model

INTRODUCTION
Today, utility computing for end-users already exists as cloud-based web applications delivered through the browser. As these applications evolve and demand richer platforms under which they run, they will soon confront and challenge not only the desktop but also the confines of the browser.

This paper proposes an exploration of new user-interface metaphors for end-users that better combine desktop computing with cloud-based services. The current desktop metaphor, developed in an era that was focused on local computation and resources before networked systems were prevalent, is an inadequate basis for incorporating such remote services. Its gestures and artifacts (e.g., devices, folders, drag and drop) heavily reflect the local file system model and its associated operations, all grounded firmly in hierarchy and physical locations. As networking and file servers became prominent, the desktop metaphor evolved very little to support sharing and access control. Instead, it has relied on location even more: mounted volumes represent additional hierarchical storage on external disks or "remote" servers, and as for access control, location often serves as a proxy for sharing-policy (e.g., designated folders for Public documents).

UIST'10, October 3-6, 2010, New York City, New York.

The central problem is that the desktop metaphor still focuses on files, hierarchy, and physical locations, which contrasts sharply with cloud-based application models. For congruence with the coming age of cloud computing, new interface metaphors should directly represent people, promote simple gestures for sharing and specifying access control, while abstracting away physical location of resources, and to a large degree, hierarchy from the user model.

Proposal
This paper proposes a user interaction model for a "social desktop", one that addresses both access control and organization of users' documents and resources. A key feature is a lightweight, user-created grouping construct, which provides a convenient mechanism to express context among a set of related resources while also implicitly specifying access control. Groups are similar to file folders, but with a few important distinctions. First, groups are not limited to holding representations of files, but also representations of other objects, such as people, applications, and hardware resources. As such, groups also serve as a natural security sandbox for the contents within. Second, they do not enforce hierarchy, allowing a particular object to exist in multiple groups. Finally, groups automatically maintain and visualize relationships with other groups containing related objects, relaxing the need for users to know exact locations to organize and retrieve their resources. Groups are considered lightweight because they are cheap (computationally and cognitively) to create, find, and modify.

The main hypothesis is that the properties described above—i.e., lightweight contexts, unenforced hierarchy, social metaphors—not only facilitate the organization and retrieval of local and remote resources, but also enable greater adherence to the *implicit security* [18] principle, whereby the intended security policy is inferred directly from the user's main task.

WALK THROUGH
Let's introduce Mary, an end-user persona representative of lay computer users, who in a few years will buy a new computer that might come pre-bundled with remote computing utility services (e.g., storage, email and phone accounts). She represents a large class of users in what she wants to accomplish using her home desktop: e.g., chat with her family, share photos, and occasionally buy goods online for her

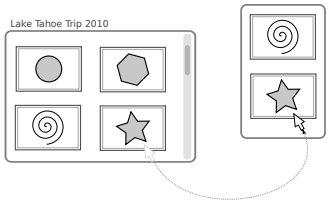

Figure 1: An object (e.g., photo) can exist in multiple groups. Here, Mary creates an unnamed group (right) to serve as her photo shortlist.

grandchildren. She isn't interested in maintaining her system nor is she likely to bother with learning how to secure it with appropriate access control policies.

Let's consider a new model for Mary's future user interface. Imagine that Mary has just returned from a family summer vacation, with a camera full of pictures. Her ultimate goal is to sort through the pictures, create an album from a select few, and share the album with her friends and family, by perhaps posting it on a photo-sharing site (e.g., Flickr). While creating the album, she might want to perform modest editing on some photos—and possibly with a little help.

First, Mary connects her camera to her computer, and image thumbnails appear automatically in a new group. The group has a default name, e.g., Images Taken with HP Camera, 6/14/10–6/19/10, but she decides to rename it to Lake Tahoe Trip 2010. Mary isn't concerned about where these images are physically stored, nor does she worry about moving the group to a "known" location (e.g., My Pictures). She takes for granted at least two things: a) that even once the group disappears from view, she will be able to easily find it again, and b) should her computer crash and need replacement, her photos remain safe (e.g., because her data is automatically transferred to the cloud shortly after connecting the camera).

Mary wants to select the best images of the entire set, so she creates a new group, which she doesn't bother to name yet, if ever, though we'll refer to it as the *shortlist* group. As she examines each source image, she decides whether to drag it to the shortlist. If she does, the thumbnail appears in the shortlist but continues to exist in the original group (Figure 1). Mary understands that both thumbnails refer to the same picture. If she wanted to, Mary could select the thumbnail in either group, and the system could show her visual hints of all other groups where she might find that image.

Once Mary is satisfied with her selection of potential photos, she closes the group containing all her photos and begins to closely review the shortlist. She realizes that some of the photos could use some cropping and brightness adjustment, but she isn't quite sure how to proceed. Mary decides to ask her favorite granddaughter, who frequently (though not always patiently) provides Mary technical support. Mary might just type in her granddaughter's name, but because the name is so common, Mary would rather just browse for her granddaughter's avatar. She first pulls up her family group, which contains a jumble of avatars representing various family members, organized visually (by Mary) in some rough manner that makes sense to her. Mary doesn't immediately

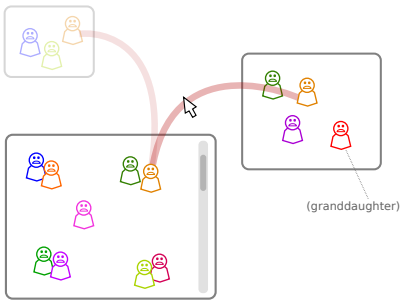

Figure 2: Users browse objects by relationships, maintained by the system. Mary can find her granddaughter by selecting the granddaughter's mother (which happens to be convenient). The system produces visual hints to guide Mary to groups that contain the mother, one of which leads to Mary's granddaughter.

see her granddaughter, but she does see her granddaughter's mother, so she selects the mother, which visually hints at other groups to which the mother belongs. Mary follows one of the hints, opens the group containing her granddaughter's nuclear family, and sees her granddaughter's face (Figure 2). Mary could click on her granddaughter to see visual hints for groups that might contain others related to her granddaughter, as well as groups that contain documents Mary currently shares with her granddaughter.

A visual indicator suggests that Mary's granddaughter is available, so old-fashioned Mary decides to call. She drags a telephone icon, representing a service, onto her granddaughter (or vice versa). Mary asks how she might edit her photos, to which her granddaughter recommends a popular free service, PhotoChop. Mary knows of a directory to find services, but her granddaughter sends her an instant message, containing an icon of PhotoChop (which Mary might drag to a place of her choosing, such as a favorites dock), and bids Mary farewell. PhotoChop might be a web application with an advanced interface, a downloadable program, or some hybrid.

Mary creates a new group, which we will refer to as *touch-up*, and from the shortlist, throws in photos that need editing. She drags the touch-up group onto PhotoChop, knowing that PhotoChop can only access photos in touch-up, reminded by a visual link between the touch-up group and the window

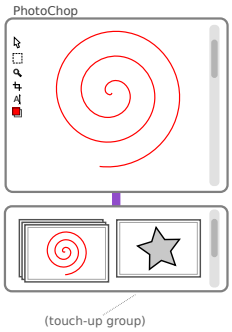

Figure 3: A service (e.g., the image editor) can only access contents within associated groups. Mary starts the editor by dragging the touch-up group onto the service's icon, implicitly specifying a security sandbox.

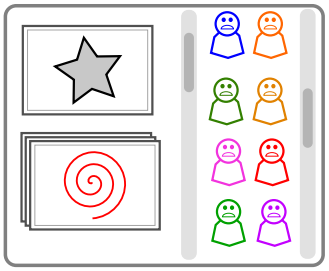

Figure 4: Users grant access to resources by dragging peers into the group containing those resources.

containing PhotoChop (Figure 3). Eventually, she gets stuck while attempting to change the brightness of a photo and finds that she's made quite a mess. She locates her granddaughter's avatar and drags it into the touch-up group to share the group (and photos) collaboratively with her granddaughter. Mary explains the problem, prompting her granddaughter to assist. Mary feels secure that should she not approve, she hasn't lost any of her original work because the system automatically maintains revisions.

Mary is pleased with all of her touch-ups, and closes the window and associated touch-up group. Before Mary posts the photos, she would like feedback from her family. She finds each family member (or groups with family members) and drags them into the shortlist to share her photos (Figure 4).

Days later, Mary decides to post her shortlist on a public photo-sharing service. Mary has lost track of her shortlist group, but she can find it again by a) either selecting her HP camera or any of her family members with access to the shortlist, and b) subsequently following group relation hints—or searching at any point along the way. She orders her photos accordingly, might add captions, and drags the group onto the photo-sharing service. A window appears, containing a service-specific web page, with various forms already pre-filled for Mary to confirm and submit (Figure 5). Mary might add more photos at a later date to the group, which could automatically trigger their publication.

RELATED WORK

Below summarizes previous studies on resource sharing, organization, and access control:

Sharing and Access There have been many recent studies on access controls in the context of file sharing. Representative work includes Shen et al.'s proposal [16] to extend current access control models to support new rights (e.g., beyond simple read/write) relevant to synchronous collaborative environments (e.g., rights to change content on a shared display). Cao et al. [6] propose a new approach, called intentional access management (IAM), which generates an enforceable policy based on user intention and refined through an interactive process modeled on wizards. Whalen et al.'s survey [20] suggest that users do change file access permissions over time to adjust for changing (social) contexts, while Smetters et al.'s work [17], which examines sharing access controls over a much longer evaluation period, suggests the opposite: that users don't change policies after initially set, but policies tend to be quite complex. Smetters et al.'s study indicates that a file's context (rather than content) determines

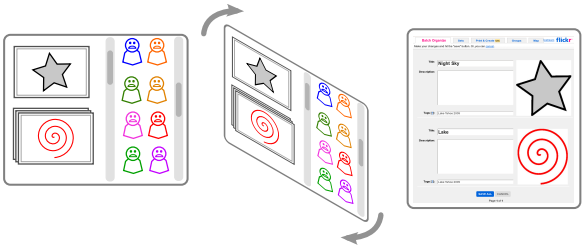

Figure 5: The system strives to integrate metaphors, e.g., by blending groups, services, and web pages. Upon dragging the photo-sharing service onto a group of photos, a new window (e.g., on the reverse of the group) displays a service-specific web page with pre-filled forms.

its sharing policy and suggests that location is often used as a proxy for context. Voida et al. [19] examine sharing in practice and note that current approaches for sharing are highly domain specific—and hence mechanisms for specifying access vary substantially.

While many of these and similar studies focus on the traditional access control list model, Voida et al. propose a visual *sharing palette*, in which users share files by dragging desktop files to icons representing peers. The approach is similar in spirit to the model explored here, though the latter extends the general idea to a wider variety of contexts (e.g., sharing hardware resources and file organization).

Non-hierarchy, Links, and Search There is evidence that non-hierarchy is an idea (still) worth further investigation. The Lifestreams project [7] articulates the idea of on-demand folders, or views, that are automatically created for the user to encapsulate a set of files, typically through the process of search. Similar in spirit, semantic file systems (SFS) [9] create virtual directories to group files by meta-data attributes. Marsden et al. [13] argue against the usability of hierarchical file-systems, proposing instead file-system models with transactional, database semantics. As with others, Rekimoto [14] proposes that time is a better metaphor than hierarchy, for the purposes of organizing. ContactMap [21] proposes representing peers as first class objects and argues that a flat (visual) namespace for organizing them is sufficient. NHFS [8], the non-hierarchical file-system, implements a prototype user-space file-system that enables users to place unique file instances in multiple directories. LiFS [2], the linking file system, explores relaxing hierarchy by enabling user-created, named links between files. To address finding files and documents, Bergman et al. [5] demonstrated that users tend to prefer folder navigation over desktop search.

The model proposed in this paper borrows concepts from many of those projects but differs by exploring: a) on-demand, user-created contexts that may persist for grouping objects, rather than serving as temporary views, b) links to other contexts, created automatically, rather than solely manually, based on common files, attributes, or shared status with other peers, and c) a hybrid approach for finding files, based on navigational browsing bolstered with search. The first concept, user-created contexts, explores the hypothesis that when users need to group common files, the ability to create persistent groups in a lightweight manner is more natural than to a)

assume a flat namespace, and consequently b) require users to construct search queries using meta-data attributes (e.g., time, author, etc) to find files. The second concept, that of automatic linking, explores the hypothesis that while users need to group files, they may not need to (manually) organize those groups. Perhaps if users feel confident that they can easily find their files and groups in the future, they may yield control of organization to the system. The hybrid approach for finding files explores whether users can effectively search while navigating, without losing mental context. One potential differentiating feature is that the search ranking algorithm can leverage group links and distance to order results based on the current user context.

Desktop Access Control Several existing projects explore usable access controls on the desktop, beyond traditional mechanisms of requiring users to manually specify security policy per resource. Chameleon [11, 12] proposes a model that partitions the desktop into one of several security sandboxes that map to user task types (e.g., "personal", "work", "communications") to limit the effects of malware. Other projects propose similar ideas, such as formal models for role-based access controls (e.g., Sandhu et al. [15]), compartmentalized or restricted execution environments (e.g., Goldberg et al. [10]), approaches that combine information flow control with compartmentalization (e.g., MITRE's CMW [4]), and fully compartmentalized machine metaphors (e.g., WindowBox [3]), also exemplified in virtualization.

Virtual machines may be too heavyweight, and compartmentalized models may be too inflexible (at worst) or inconvenient (at best) to support easy, intended data sharing across task types. Static or predetermined role-based sandboxes might confuse users when the mapping between tasks and the appropriate sandbox is unclear or not one-to-one.

Instead, this paper proposes exploring whether lightweight contexts can also serve as a user model for specifying access controls, which holds several promising features. First, it leverages the same mechanism for grouping files, peers, system resources, and applications to define context. Second, if easily created, user-defined contexts are more fluid than static roles, and potentially more natural for complex access control policies that reflect dynamic human relationships and workflows. Finally, users need not divert from their direct task to first specify a security policy, as the system has a reasonable chance of inferring it from user-created contexts.

CONCLUSION

Conventional operating systems are ceding function to cloud-based services. This suggests that all applications may ultimately run within an evolved web browser, marginalizing the role of the OS and its local desktop model. Google's recent Chrome OS [1], with its notable absence of a desktop, suggests such a future.

There are several concerns with a web-only application environment. First, the web imposes a page (or document) metaphor—prompting developers to either force their applications to conform to that metaphor, else work harder to escape it. Second, one important role of traditional operating systems is establishing a consistent user interface paradigm. But due to the heterogeneity of the web, there is not an entity that provides and enforces a consistent interface model, inviting a fragmented user experience across web sites. We believe that a unifying model for the next generation desktop is not only necessary but should promote representations of people, their social networks, and gestures for sharing, all while de-emphasizing the importance of location.

REFERENCES

1. The Chromium Projects: Chromium OS. http://dev.chromium.org/chromium-os.
2. A. Ames, C. Maltzahn, N. Bobb, E.L. Miller, S.A. Brandt, A. Neeman, A. Hiatt, and D. Tuteja. Richer file system metadata using links and attributes. In *IEEE MSST*, 2005.
3. D. Balfanz and D. R. Simon. WindowBox: a simple security model for the connected desktop. In *USENIX Windows Systems Symposium*, 2000.
4. J. L. Berger, J. Picciotto, J. P. L. Woodward, and P. T. Cummings. Compartmented mode workstation: Prototype highlights. *IEEE Trans. on Software Eng.*, 16(6), 1990.
5. O. Bergman, R. Beyth-Marom, R. Nachmias, N. Gradovitch, and S. Whittaker. Improved search engines and navigation preference in personal information management. *ACM Trans. on Info. Systems*, 26(4), 2008.
6. X. Cao and L. Iverson. Intentional access management: making access control usable for end-users. In *SOUPS*, 2006.
7. E. Freeman and D. Gelernter. Lifestreams: a storage model for personal data. *ACM SIGMOD Record*, 25(1), 1996.
8. R. Freund. *File Systems and Usabilitythe Missing Link*. Bachelors thesis, University of Osnabruck, 2007.
9. D. Gifford, P. Jouvelot, M. Sheldon, and J. O'Toole. Semantic file systems. *SIGOPS Operating Systems Review*, 25(5), 1991.
10. I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A secure environment for untrusted helper applications confining the wily hacker. In *USENIX Security Symposium*, 1996.
11. A.C. Long and C. Moskowitz. Simple desktop security with chameleon. In *Security and Usability*. O'Reilly, 2005.
12. A.C. Long, C. Moskowitz, and G. Ganger. A prototype user interface for coarse-grained desktop access control. *CMU Technical Report, CMU-CS-03-200*, 24, 2003.
13. G. Marsden and D. E. Cairns. Improving the usability of the hierarchical file system. In *SAICSIT*, 2003.
14. J. Rekimoto. Time-machine computing: a time-centric approach for the information environment. In *ACM UIST*, 1999.
15. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2), 1996.
16. H. Shen and P. Dewan. Access control for collaborative environments. In *ACM CSCW*, 1992.
17. D. K. Smetters and Nathan Good. How users use access control. In *ACM SOUPS*, 2009.
18. D. K. Smetters and R. E. Grinter. Moving from the design of usable security technologies to the design of useful secure applications. In *NSPW*, 2002.
19. S. Voida, W. Edwards, M. Newman, R. Grinter, and N. Ducheneaut. Share and share alike: exploring the user interface affordances of file sharing. In *ACM CHI*, 2006.
20. T. Whalen, D. Smetters, and E. Churchill. User experiences with sharing and access control. In *ACM CHI*, 2006.
21. S. Whittaker, Q. Jones, B. Nardi, M. Creech, L. Terveen, E. Isaacs, and J. Hainsworth. ContactMap: organizing communication in a social desktop. *ACM TOCHI*, 11(4), 2004.