It's All the Same to Me: Data Unification in Personal Information Management

David R. Karger[*] William Jones[†]
MIT Computer Science and AI Laboratory

February 20, 2006

1 Introduction

Information fragmentation is a pervasive problem which is felt in several stages of personal information management (PIM) [ref]. As the example in the introduction to this special issue on PIM illustrates, even a seemingly simple decision, such as whether to say "yes" to an invitation, often depends upon a number of different kinds of information - information from a calendar, from a paper flyer, from web sites, from a previous email conversation, etc. Such information can be fragmented by physical location—some information, for example, may be on a laptop computer we use at home, other information may be on a desktop computer we use at work and one or more PDA or smart phones. But even on a single device, information is often fragmented by the very tools that have been designed to help us manage our information. Applications often store their data in their own particular locations and representations, inaccessible to other applications. Digital documents and other files are managed by the by a file manager, email messages by an email client, bookmarks to web sites by a web browsers, and so on. New applications such as Microsoft OneNote [reference] introduce still more management tools with little or no integration to previous forms. This leads to numerous problems.

Broken linkages. Much of interest about our information is its connection to other pieces of information. For example, we like to know that a given individual is the author of a given document, or that a given email message is relevant to a particular appointment. If these pieces of information are managed by different applications, we may find that none of them is able to properly record or present the linkage to the information they do not manage. We may need to annotate the relationship manually, in a comment field not suited to recording it. Later, we may need to perform a difficult search in order to access another representation of information we are already looking at—this presuming in the first place that the user remembers which applications manage the desired information!

Partitioned organizations. Each application tends to offer organizational tools for it own, and only its own, information. Such organizational fragmentation creates problems in everyday PIM actions such as keeping and finding. As a result we may sometimes need to look in several places, physical and virtual, in order to gather together the information we need for a particular task. We may also be less certain where and how to keep newly encountered information[reference Marshall & Jones article in the issue]. Or do "have it" already? If we keep the information again anyway ("just in case") we may then face some serious problems with consistency and updating later on. People can rightly complain that they have "too many hierarchies" [Boardman et al. 2003; ravasio, 2004] and people sometimes go to great lengths to bring their information together into a single organization whether based in files, paper or email messages [jones 2002].

[*]M.I.T. Laboratory for Computer Science, 545 Technology Square, Cambridge MA 02139. E-mail: `karger@theory.csail.mit.edu`. URL: `http://theory.csil.mit.edu/~karger`

[†]

Redundant representations. Our desire to span application boundaries in organizing or linking information often forces us to repeat the same data in several places. For example, a name such as "Jill Johnson" might appear in an address book, and also as the creator of a photograph in a digital photo album. Changes to one version of the data (a new married name, for example) often do not propagate to other versions of the data. Also, we may experience the frustration of having some operations - name resolution, for example - available in one place (when sending email) but not in another (when working with photographs).

In a study of users' desktop environments, Ravasio et al. [2] observed that users are themselves aware of this problem with fragmentation. They state that "the systematic separation of files, emails and bookmarks-was determined by three users to be inconvenient for their work. From their points of view, all their pieces of data formed one single body of information and the existing separation only complicated procedures like data backup, switching from old to new computers, and even searching for a specific piece of information. They also noted that this separation led to unwanted redundancy in the various storage locations."

If the computer has been an unintended agent of information fragmentation, it can also be used to help us "put the pieces together" again. This article provides a sampling of some of the ways in which our personal information might be better integrated. Underlying, requisite elements of an integration such as shared protocols, address spaces and interchange formats are then briefly discussed. The article concludes with a look at research prototypes that illustrate varieties of approach to the fundamental challenge of personal information integration.

2 Unification Methods

In this section, we will outline several approaches to information unification. Each approach relies on some "common denominator"—something that is the same about all the information to be unified—and ignores other, irrelevant aspects of the information. Doing so lets us operate on all of the information in the same, unified way, so long as the operations rely only on the common representation. It also means that all of our tools can work with the same information, so long as they make the effort to understand the common representation.

The idea of a common denominator is is in tension with applications' desires to have rich, specialized representations of their content. Without those, it is not possible to offer powerful domain-specific operations. The tradeoff, of course, is that the shared representation lets applications interact with information from outside their domains, leading to unification.

2.1 Visual Unification

Perhaps the most universal and most shallow unification is at the level of the display. In today's GUI environments, nearly every application launches a "window" within which its information is displayed. The common denominator here is the representation by a rectangular array of pixels. Relying only on this common representation we can resize, move, hide and reveal different information objects, regardless of their types or managing applications.

Bringing up several application windows at once offers us the opportunity to view several information objects simultaneously. This lets us look for recurring patterns among and important connections between the items in view. As cognitive psychologists know, our view of items can act as a powerful extension to our limited internal working memory for information.

Unfortunately, as we attempt to arrange information on a computer display, we experience problems. For example, applications involved in rendering the items of a collection may each consume a large part of the display. Documents, email messages, web pages, etc. may each "live" in a large window with attendant menus, toolbars, jumping-off points and default presentations. Because the window manager treats the application opaquely as a rectangle full of pixels, it cannot select the one piece of the display that the user actually cares about. A common consequence is *window clutter*—a display filled with windows, often obscuring each other and each competing for our attention. The information we lay out in order to see and understand can turn into a jumble that actually impedes our ability to work effectively.

Tan et al. [3] have tried to address this problem by developing *wincuts.* The idea of wincuts is that a user can select a (rectangular) sub-region of an application's window, "cut" it out of the application, and "paste" it into an entirely different region of the display. The user can then close the original application window. The wincut is alive: the user interacts with it as if with that piece of the application window. Reducing the screen space occupied by one application means that more applications can be visually unified amid less distracting irrelevant information.

While Wincuts offers some help with window clutter, it retains the other major drawback of display-level unification. Since only the display, and not the underlying data model, is unified, information must still be managed by the application responsible for it. Wincuts does not offer expanded opportunities to pass data from one application to another, or to create machine-usable linkages or shared organization structures between data from multiple applications. The fact that one can display a person in an address book and, at the same time, display a photograph of that person in a photo album offers no guarantee that one will be able to use either application to associate the person with their photo.

2.2 Files

The file system has been another, highly successful data unifier. At the most basic level, the file system offers yet another common denominator: the file. All applications can make use of the notion of a sequence of bits that can be read from, copied, or written to (although of course, writing bits into a file whose meaning is not known will serve only to corrupt that file). An email application can "attach" a file for delivery to another computer without any understanding of which application created the file or what its contents mean.

In addition, exposing all data through files means that we can write a single tool, the file system, for managing that data in a unified fashion. Using the now-standard model of hierarchical directories or folders, a user can organize files according to whatever principle they find useful, regardless of file type.

On the downside, the file system assumes so little about the meaning of the files it holds that any significant manipulation of any file requires launching an appropriate application. This will take a user away from the directory view of all files relevant to a given task, and back to an application view that shows only some of the information they want to work with. Additionally, files are typically large. An address book, for example, will usually be stored as a single file, rather than as one file per entry. This precludes using the file system for fine-grained organization. A user will not be able to put, into a directory aimed at writing a particular paper, the address-book entries of all his co-authors. He can certainly place a link to the address book, but from within that directory he would need to launch the address book and then go through an address-book specific process to find the co-authors in the address book.

2.3 Text

A significant portion of the data managed by many applications is text. It may come formatted in many different ways: within HTML documents on the web, in bulleted lists in Microsoft Powerpoint, laid out in a word processing document, or typeset in Adobe PDF. But strip away the formatting, and one is left with a sequence of characters. Most applications are able to offer text to and accept text from other applications—typically with the assistance of a clipboard application with which they all communicate. This cut and paste facility provides an intuitive way of moving data from one application to another—although the transfer is usually text-only.

Other facilities for manipulating text are still provided piecemeal. For example, in ways analogous to those we use when marking up a paper document, one can highlight and annotate selected text for a document rendered in Microsoft Word or, in a slightly different way, for a document displayed in Adobe Acrobat. However, it is not possible to perform similar operations on the selected text of an email message displayed in Microsoft Outlook nor is it possible to highlight the selected text of a Web page presented in the Microsoft Internet Explorer[replace with non-Microsoft examples]. Even basic cut and paste is sometimes omitted: for example, it is often impossible to select and copy the text of an error message that has popped up on the screen. Another barrier appears on the web, where a desire for fancy views often leads sites to present headers and such as images containing text that cannot be selected.

The common model of text also means that most applications can contribute some (stripped down) representation of their data to a *text search engine.* This gives users a well-understood framework for searching by content over all of their data, independent of which application owns any given piece. Significant buzz has developed around the recent set of "desktop search engines" such as Google desktop MSN desktop, SIS, and Enfish, that offer this capability.

Support for such integrative text searching is now finding its way into the operating system in new releases of both the Macintosh OS and Microsoft Windows.

technique	offered by	operations	enables
standard datatype	text, files	cut/copy/paste	unified searching, data transfer
unified presentation	window manager, wincuts	layout, tile, show, hide	simultaneous view of information
naming		refer, dereference	list below
grouping	directories, Taskmaster, Presto,	add/remove	organization, browse, simultaneous view
cross-reference	OLE, COM web	embed, traverse	simultaneous view, orienteer
attributes	Presto, XML	annotate query	annotate, search organize, browse
relations	RDF, Haystack		record relationships unified search orienteering

Figure 1: Different unification techniques are offered by different software systems. Each unification supports different types of operations and enables different data management activities by the end user.

3 What's in a Name?

We now devote an entire section to what might seem the most prosaic of all common denominators—the idea of that an information object can be named and therefore referred to. Names are at the core of the filesystem (file identifiers) and the Web (URLs). Names require no understanding whatsoever of the opaque objects that they name, while offering a tremendous number of operations that are crucial to supporting users working effectively with their information, including grouping, annotation, and linking. Sadly, although working with names is theoretically quite easy, most applications currently maintain their own namespaces for their objects, and do not expose these names to reference by other applications. Thus, a significant opportunity for information unification is missed.

3.1 Grouping

Once objects are named, they can be grouped by the users into meaningful collections. This gives users the opportunity to gather all the information objects they wish to use together. Frequently, we group information relating to a particular task. For example, we might group information concerning hotels in a city in order to select a hotel for our stay in that city. Traditional file directories provide one means for grouping information together, so long as the information objects are files.

Directories may provide other valuable grouping functions as well. In one study, for example, user-created folder structures sometimes appeared to serve as a problem decomposition or project plan [jones et al. 2005]. People also reported that their folder structures gave them an important sense of control over their information and helped them to "see" their information better.

Unfortunately, despite the fact that they rely only on being able to refer to their contained items, different applications insist on managing collections of their own information—files go into file folders, email messages into email folders, web references into bookmark folders accessed through a web browser, and address book entries into address book folders.

would like to discuss these two paragraphs but this may be out of scope—key of this document is that collections are doable, not to discuss the best way to do them Research has explored the more general and flexible notion of a collection[italicize][Dourish et al, quan & Karger]. Items can be manually assigned to a collection (e.g. files placed into a folder) but items (or at least suggested items) for a collection can also be generated based upon a match between items and a "definition" (e.g. a query) for the collection. Limited support for the automated creation of collections is available now via features such as Microsoft Outlook's "Search Folders". Variations on this are now expected in new releases of both the Macintosh and Microsoft Windows operation systems [ref].

Even within the same folder organization competing organizational schemes may suffer an uneasy co-existence with each other. People may apply one scheme on one day and another scheme the day after. There may, for example, be a tension between organizing files (images, articles, etc.) by project for current use and organizing these same files by content for repeated re-use [jones et al.]. Although research indicates that, given the right support, people are able to assign multiple categories to an information item [quan, karger,], the support available to the average computer user (e.g., for creation of shortcuts in Microsoft Windows or for aliases on the Macintosh) remains quite primitive.

3.2 Annotation

If an object has a name then it can be annotated. Annotations let us attach metadata to objects; metadata that can be useful for organizating, locating, or simply describing those objects. Files have creators and creation dates; media files have genres and bit-rates, mail messages have recipients and subjects, and appointments have times and places. Collections, too, can be usefully annotated. For example, if a collection of information relates to a task ("Find a hotel"), then it may be useful to assign task-like properties like "remind by" and "due by" ([jones et al.]) which might then appear as appointments in an electronic calendar or trigger a reminder (via pop-up or email message) later on.

As with collections, although metadata offers a useful, cross-application unification opportunity, we are currently hampered by the lack of a common representations for the names of objects we wish to annotate and the metadata that we wish to record. Thus, file creation times and creators are stored in the file system and navigated on the desktop, while mail message creation times and creators are stored in email applications, and music composers and composition dates are stored in ID3 tags in media files. This generally means one must launch a specific application to view or search by a particular type of object. Until recently, such a simple task as "finding the stuff I created last tuesday" required launching multiple applications. The recent crop of desktop search tools attempts to extract metadata from many different file formats to support integrated searching over that metadata, but the lack of a standard metadata representation means that you are out of luck if no translator has been written for your application.

may also be out of scope Another problem with metadata is getting it into the system in the first place. The time of our last interaction with a document (email message, web page) is recorded currently. But many other aspects of the interactive context are not. For example, as we create a new document, send an email message or navigate to a web page, we may have a particular task in mind, but there is very little support communicating this task to the computer. Newly created documents, for example, are often placed, by default, in a place like "My Documents". In general, the context we "share" with the computer in our interactions with information items, which could be very useful for organizing or retrieving them later, is very limited.

3.3 Linking

A third use of references is to link information objects together. Such a link can be used to record various types of relationships between the information objects. In a sense both grouping and annotation are a special kind of linking—from a collection to a containing item, and from an item to a piece of metadata—but there is also great value in directly linking two distinct information objects. One is to use the linked objects as metadata for searches, describing an object by the objects it is linked to.

As another motivation for linking, we consider recent work highlighting users' preferences for finding information by *orienteering* [**?**]. Rather than jumping directly to needed information, users often try to locate it by starting with a known object and taking repeated navigation steps to related objects, aiming to home in on the desired information. We use this approach frequently when navigating the web, or when seeking files in our directory hierarchies. To support such orienteering, it is necessary to connect information objects to other related ones. When data is fragmented, it may be impossible to record some of those linkages, meaning that they will be absent when the user needs them for orienteering. Ravasio et al. indicate that users are aware of their desire for linking: “most interviewees expressed the need to have their information linked together (e.g. article author and respective address book entry, or citation and cited article, etc.) and in general, to have more content-based and context-based access to their information [2].”

Another approach, typified by Microsoft’s OLE and COM frameworks, offers the ability to launch an application to handle the named object, and embed that application’s view inside that of the containing application. In the same way, images on the web are often displayed embedded in a web page, instead of as links that can launch a new application. Various other data types such as flash can similarly be embedded if appropriate plugin applications are installed with the web browser. Embedding is useful in that it gives users information about the related object without require them to navigate away from their exiting context.

The most recent big success of this linking approach was the World Wide Web. One of the most important contributions of the WWW was to define a single, shared URL namespace that lets users craft names for arbitrary web objects. By placing references to those objects in other web pages, authors give users the ability to navigate smoothly from object to related object. The embedded names can refer to objects that a web browser could not interpret; this failure becomes apparent only if the user chooses to actually navigate to the named object. This works quite well in an environment where some users have and other have not installed plug-ins and extensions to handle a variety of new object types. Users can see references to and discussions of the named objects even without the extensions, and can then make individual choices about whether it is worth the work of extending their own environment to handle the new object type.

4 Looking Forward

Having discussed a number of currrent approaches to unification, and the benefits of them, we now look ahead to some other approaches currently on or beyond the horizon.

4.1 XML

XML is a rapidly spreading common representation for hierarchically structured information. Much as Wincuts tries to break up monolithic application windows into small pieces so that the user can get the one piece they need, XML aims to break up monolithic applications files into a finer grained collection of information objects that can be manipulated by multiple applications. Whereas a traditional word processing program would store its documents as an opaque file, an XML representation of such a document can use a standard syntax to expose, as separate fragments within the file, the author of the document, its title, and its individual sections. XML thus offers a possible common syntax for representing metadata about its information objects, thus supporting a unified approach to annotation. The common representation of individual fragments offers the opportunity for reference to those individual fragments (for grouping or linking). For example, were an address book to store its data in XML, it might become possible to place individual address book entries into a bookmarks folder.

4.2 Taskmaster

Bellotti et al. [1] performed field studies that demonstrated how end users have shoehorned a substantial portion of their data inside their email clients—a kind of ad hoc unification. Bellotti et al. then developed a new email client, Taskmaster, with this usage in mind, offering a more effective unified environment for end users managing information. Taskmaster is the subject of another article in this collection, so we give it short shrift here. Taskmaster places users in an email management context but lets them group arbitrary objects, not just email messages, into “thrasks” (threaded tasks). In addition to grouping, Taskmaster allows that user to attach metadata such as “deadline” or “action” to each information object, regardless of type, and to use these attributes to organize and retrieve information in the system.

4.3 The Universal Labeler

The Universal Labeler (UL) pursues a unification with respect to access and organization through its support for a simple "Label With" feature. Label With is a variation on the Save As dialog that allows people to use their file folder structures as a more general classification scheme for the organization not only of e-documents and other files but also email messages and web pages.

The UL takes its attempted elevation of the everyday file folder a step further through its "Task Management" feature. With Task Management users can set "Remind me by" properties and a "Due by" properties for any selected folder. Properties are represented as special appointments in the Microsoft Outlook calendar. In this way, Task Management supports a rudimentary unification of information and task/time management.

The UL also includes a project Planner module which, like Label With and Task Management, works as an extension to a person's file folder hierarchy. The Planner provides a rich-text overview for any selected folder hierarchy which looks much like the outline view of Microsoft Word. A hierarchy of folders appears as a hierarchy of headings. The view enables users to work with a folder hierarchy just as they would work with an outline: As headings are added, moved or deleted, corresponding changes are made to the folder hierarchy - the Planner is simply another view into the file folder hierarchy. But the Planner also provides document-like features not available in a standard file manager:

- Support for a"drag and link" action of excerpting. It is possible to select text of interest, drag (or copy) and then drop (or paste) into a project plan. A link to the source of the drag (or copy) is automatically created. Often we are mainly interested in only a small part - a name, number or phrase - of the email message, web page, or e-document that we are reading. But we might like to return to the rest of the information item later on.

- Support for"create and link". A link to the newly created information item (e-document, email message) is automatically created at the insertion point in a project plan.

- Ordering of elements. Users can order headings, subheadings and links of a project plan however they like. People depend on ordering as a way to establish priorities and to direct their attention to "first things first".

- Notes. Users can include notes just as they would in a document. Notes can, for example, be used to provide clarification for an associated heading.

Behind the scenes, the Planner is able to support its more document-like outline view by distributing XML fragments as hidden files, one per file folder, which contain information concerning notes, links and ordering for the folder. The Planner assembles fragments on demand to present a coherent project plan view including notes, excerpts, links and an ordering of subfolders (and sub-subfolders).

4.4 Haystack

The Haystack project [**?**] represents an effort to offer a unified data environment *ab initio*. Haystack takes the notions of grouping, annotating, and linking opaque information objects to a much finer grain than files. Even inside individual applications, much of the information management reflects the creation and usage of (binary) relationships connected information objects. For example, people are a data type appearing in many various applications that manage their relationships to email messages (as senders in an email application) to music (as composers in a jukebox program) and to appointments (as people to meet in a calendar program). For some users (in the entertainment industry) those sets of people might overlap. It is therefore worth treating the individual people as first-class information objects that can themselves be annotated, organized, and linked. Haystack takes this approach to its limit, attempting to name an annotate every possible individual information object.

As was discussed above, the only real prerequisite for such an approach is to give each information object a unique identifying name. Haystack uses RDF, an emerging World Wide Web standard model for naming information objects and for recording relationships about those objects. In RDF, any object can be given a Unique Resource Identifier (with a syntax similar to URLs), and any two information objects can be linked by connecting their URIs with a *statement* naming (in a machine readable fashion) the relationship between them. In the Haystack data model, a typical file will be shredded into many individual information objects of various types connected by application-specific relationships. RDF can be stored as XML files or managed within a database.

Haystack's user interface is responsible for taking these many small objects and assembling them into traditional-looking information displays. But since each information object in view in, say, an email management interface is itself a distinct entity in the data model, a Haystack user can click on anything in view and immediately "navigate to" the clicked object in order to get more information about it. It is also possible to invoke *all* relevant operations on an object any time the object is in view; not just when it is being presented by one particular application.

Once individual information items are "named" at a fine grain for the purposes of grouping them, it also becomes possible to consider linking them by various relationships and annotations. Haystack gives the user a web-like navigation paradigm: by clicking on (say) the author of the message, the user can navigate to a view of that author (which is constructed by looking up important objects related to that person, and laying them out in the style of an address book). objects together. Users can benefit from being able to orienteer from a document to one of the authors of that document, to a photograph of that person, to a representation of the location where that photograph was taken, to a map of that location, and so on. Similarly, traditional drag-and-drop operations can be applied by users to create collections of related objects, or to create annotations linking information objects together.

References

[1] Victoria Bellotti, Nicolas Ducheneaut, Mark Howard, and Ian Smith. Taking email to task: the design and evaluation of a task management centered email tool. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352, New York, NY, USA, 2003. ACM Press.

[2] Pamela Ravasio, Sissel Guttormsen Schär, and Helmut Krueger. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact.*, 11(2):156–180, 2004.

[3] D. S. Tan, B. Meyers, and Mary Czerwinski. WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In *Extended Abstracts of Proceedings of ACM Human Factors in Computing Systems CHI 2004*, pages 1525–1528, 2004.