A Personal Information Management Scheme Using Shared Labels and Implication Links

Gaurav Dubey  
Information Sciences and Technology  
Penn State University  
University Park PA 16802  
gxd176@psu.edu

Xiaolong (Luke) Zhang  
Information Sciences and Technology  
Penn State University  
University Park PA 16802  
lzhang@ist.psu.edu

ABSTRACT
The tree-based file management scheme is universally deployed as a personal information management approach; however, it has several weaknesses. In this paper, we identify and discuss some of the weaknesses of the tree-based approach: these include conceptual fragmentation, conceptual non-comprehensiveness, entry point restriction, and enforced ordering. After describing some previous attempts at building alternative schemes for PIM, we propose our own scheme, based on shared labels and implication links. Location addresses consist of unordered sets of labels, and each location has connections to other addresses that are subsets and supersets of the location’s address. Implication links can be used to capture implication relationships between concepts: specifying a link from one label to another means that the first label will automatically be accompanied by the second in addresses.

We discuss the components of the new scheme, in addition to methods for manipulating it. We describe the ways in which the new scheme addresses the problems identified with the tree-based approach. We also present a mockup of a prototype application for browsing information repositories based on the new scheme. To conclude, we discuss future research directions, including theoretical analysis, implementation, and testing.

INTRODUCTION
In this day and age, computer users possess thousands of files spread across multiple devices and locations, representing hundreds of gigabytes of data. However, the abstract structure used to manage this data has been relatively unchanged since its introduction. The tree-based approach to personal information management (PIM) in computer systems has been around for decades: one of the earliest descriptions of it is by Daley and Neumann, as implemented in the Multics operating system (Daley & Neumann, 1965).

The majority of modern operating systems and file management systems use some version of the tree-based approach. However, the amount of data users possess has grown tremendously over the years, and users of existing tools often have great difficulty finding the files they need (as described by Jones, 2007, for instance).

Despite the development of effective desktop search engines, many users have reasons why they still use folders (Jones, Phuwanartnurak, Gill, & Bruce, 2005). Clearly, the existence of some sort of navigable structure is useful. However, we have identified a number of problems with the tree-based approach. In this paper, we describe those problems, and we propose a new PIM scheme to address them. The new scheme uses shared labels and implication links to construct a more richly connected information repository, where shared labels form connections between conceptually related locations. We also present a mockup of a prototype application for accessing and manipulating repositories that make use of the new scheme. (We use the term information repository to refer to a logically and functionally contiguous storage space. Examples include an e-mail account, the filesystem on a personal computer, or a web browser’s bookmark collection. In terms of granularity, an information repository lies between a personal space of information, defined by Jones as ‘everything informational related to’ a user, and a personal information collection, defined as a ‘personally managed subset of a PSI’ (Jones, 2012).)

RELATED WORK
There have been a number of attempts to create superior alternatives to the tree-based approach. Relevant work includes alternative data structures, such as multitrees (Furnas & Zacks, 1994) and polyarchies (Robertson, Cameron, Czerwinski, & Robbins, 2002). In the specific domain of PIM, however, most of the alternatives we found are based on multiple categorization using tags.

These alternatives include work by Quan et al on a multiple categorization-based bookmark management tool for Internet Explorer, as part of the Haystack project (Quan, Bakshi, Huynh, & Karger, 2003). This tool allows users to create and select any number of categories while storing a bookmark. During bookmark retrieval, the tool dynamically generates a hierarchical view, presenting other intersecting categories in addition to bookmarks stored under the current intersection of categories. The interface is designed in order to closely re-
The File and Concept Browser (FCB) (Sajedi, Afzali, & Zabardast, 2012) is another PIM tool that uses multiple categorization. Categories, or concepts, are stored in a conventional hierarchical structure. However, files do not belong exclusively to a single concept—they can be members of multiple concepts simultaneously. Depending on the display mode, the tool displays the files belonging to one concept, the union of the concepts selected by the user, or the intersection of the selected concepts.

These tools all address the issues with the tree-based approach. However, in our scheme, each data item has a single location in the information repository. Labels in our scheme do not act as attributes or metadata; instead, they form the structure of the information repository. We chose this approach in order to give the user a stronger sense of there being a concrete underlying structure, while avoiding the inflexibility of the tree-based approach.

PROBLEMS WITH THE TREE-BASED APPROACH

We have identified the following problems with the tree-based approach. The new PIM scheme we propose is designed to address these problems.

Conceptual fragmentation

The names given to folders often reflect a certain concept that is used to organize the files and folders within (eg. Research, Music, LaTeX, 2010). As seen with the last example, this concept need not be unique to a single collection of items—a user could have pictures, research papers, and tax forms from the year 2010, all in separate parts of her information repository.

Under the tree-based approach, folders corresponding to a given concept could be scattered across the repository, with no connection indicated between them. This is a phenomenon we refer to as conceptual fragmentation. For instance, from within the folder ...\Documents\ASIS&T\LaTeX, there is no indication that there is another LaTex folder at ...\Documents\UIST\LaTeX, and there is no direct navigational link to this folder (or any other folder with ‘LaTeX’ as a part of its address). In the current example, the user might want to copy content from a LaTeX source file she wrote for a previous paper. Under the current scheme, however, she cannot use LaTeX as a link between two locations.

She must instead navigate to the required folder some other way, which could be a longer, more disruptive interaction.

Conceptual fragmentation is also described as a consequence of folder hierarchies in (Jones et al., 2005).

Conceptual non-comprehensiveness

Conceptual fragmentation causes another problem with tree-based information repositories: when searching for locations that correspond to a certain concept, the user cannot be sure she has accounted for all relevant locations. There might be a LaTeX folder containing important content in a part of the repository that the user missed. However, none of the other LaTeX folders provide any indication of this folder’s existence.

(Henderson, 2011) discusses the issue of document duplication, where multiple copies of the same document inadvertently end up scattered across the user’s information repository. Users often have difficulty accounting for these copies, and might end up missing or forgetting some. Conceptual non-comprehensiveness can be considered a similar issue, occurring with folders instead of files, where the user cannot account for all the occurrences of a folder name across the information repository.

Entry point restriction

With current implementations of the tree-based approach, the user must choose from a predefined set of entry points in order to start her navigation. (For instance, the Windows Start Menu has links to Documents, Pictures, Music, etc). There is some flexibility to this: for instance, the Windows Explorer Jump List allows users to access frequently accessed, recently accessed, and user-specified folders. However, this is not always implemented effectively—as an example, the pinned items in Explorer’s jump list are not synchronized with the Favorites list within Windows Explorer itself. Having to manage multiple lists may make users less likely to use this feature to its full potential.

If the user could simply specify an arbitrary starting point directly, this wouldn’t be a problem. However, this is difficult, because the user must precisely remember the address, including the specific order of folders (see the next section for more on this), along with a number of system-related terms with no particular significance or meaning to the user (such as C:\Users\Gaurav Dubey\...).

Enforced ordering

Often, while building structures to organize their information, users choose to express multiple levels of categorization. For instance, the user might categorize her paper-writing materials by conference, by year, and by material type (diagrams, LaTeX source files, etc). Under the tree-based approach, the user would have to choose which of these divisions to make first (closest to the root). Should she use ...\ASIS&T\2012\LaTeX and ...\ASIS&T\2012\Diagrams? Or ...\LaTeX \ASIS&T\2012 and ...\Diagrams\ASIS&T\2012?

The user may or may not have reasons to prefer one arrangement over the other (as illustrated in (Henderson, 2005), for
instance). However, even if she has no particular preference, she must still choose an approach. Once this choice is made, it has a strong effect on how the user accesses her data, since she is now restricted to specifying the components of a location in a specific order – one that may or may not match the order in which she recalls those components. This could cause the user to get momentarily stuck. For example, in the first arrangement above, if the user wants to access a particular \LaTeX file, it might take her a few seconds to remember what upper-level folder she needs to enter, since there isn’t a \LaTeX folder directly under her Documents folder. (Of course, she might get even more confused if there is a \LaTeX folder that simply happens not to contain the file in question.)

Enforced ordering is also discussed as a consequence of the tree-based approach in (Quan et al., 2003).

**APPLYING SEARCH**

One could argue that a search engine would also solve the problems listed above. Performing a search query for a desired folder name could bring up a list of all locations with that folder in their path. This would address conceptual fragmentation, since it would provide direct access to any folders that shared the specified concept; furthermore, as long as the search results included all possible locations, it would also address conceptual non-comprehensiveness. The ability to enter an arbitrary search query from any location in the information repository would address entry point restriction; and the ability to enter search terms in any order would address enforced ordering, at least while accessing structure (if not while creating it).

We believe that search engines in their current form are an acceptable solution to entry point restriction and enforced ordering. However, using search engines to address conceptual fragmentation and conceptual non-comprehensiveness is more problematic. There are certain superficial reasons for this – for instance, the way in which search results are ordered is not always optimal for displaying comprehensive lists of conceptually related folders, because several other files might also be displayed, and the folders might not be grouped together. Furthermore, invoking a search engine to find the folders related to a given folder could be a workflow interruption under certain circumstances – for instance, if the user wants to find folders related to the current folder, she must type a query (or copy and paste it) into a search box.

At a more fundamental level, however, lists of search results are not an integral part of the abstract structure of a tree-based information repository. Furthermore, these lists are ephemeral – they usually do not have permanence in the way that a folder does. A user might want a particular list of related locations to be a permanent navigational fixture, for convenience’s sake. However, this would result in two separate types of location – folders, and lists of related folders. How would these lists be managed? Would they be stored in the tree structure itself, or would they be stored outside it? Would the contents of these lists only include folders, or could they also include other lists? Would there be a separate organizational scheme for managing the lists of search results themselves?

Although it might be possible to make such a solution work, we consider it to be overly complicated and inconsistent. We prefer the idea of developing a new PIM scheme from the ground up – one that allows for a much richer web of connections between related locations than what is possible with the tree-based approach. We believe that our new PIM scheme addresses the problems listed in the previous section, while providing the user with an underlying structure that is consistent and straightforward.

**A NEW PIM SCHEME**

This section will describe the new PIM scheme designed to address the problems with the tree-based approach. It will describe the components of an information repository under the new scheme, as well as the issues involved in manipulating a repository.

Figure 1. The components of the new scheme. The large rectangles are locations, while the inner rectangles are labels. The central location is [ASIS&T, Research, 2012]. The arrows labeled (1) are connections to sublocations, while those labeled (2) are connections to superlocations. The arrow labeled (3) represents an implication link: ASIS&T $\rightarrow$ Research.

**Components**

These are the components that make up an information repository.

**Labels**

Labels (eg. Research, Pictures, 2012) are the fundamental components of location addresses. Labels are unique, and have universal scope across the repository. Labels are not the same as metadata or tags: where tags can be considered filters operating on a (possibly structureless) repository of items (as described in (McGuffin & Schraefel, 2004), for instance), labels are what produce an information repository’s structure. In its current form, the new PIM scheme does not use tags. (Directly applying and removing labels to and from items is an interaction method that may be studied in future research.) Note that labels are not created or deleted directly. (See the ‘Manipulation’ section for more details.)

**Locations**

Locations are where items are stored. Each location has a unique address: this is an unordered set of user-specified labels (eg. [Research, ASIS&T, 2013, Images]). Each data item is stored in exactly one location.
When describing the relationships between locations, the following terms also apply:

- **Sublocation**: a location whose address is a subset of a given location’s address. (Eg. [2012] is a sublocation of [2012, Research].)

- **Superlocation**: a location whose address is a superset of a given location’s address. (Eg. [2012, Research, ASIS&T] is a superlocation of [2012, Research].)

**Connections**

Each location has two lists of connections: a list of connections to sublocations, and a list of connections to superlocations. Connections allow users to navigate from one location to another. Any two locations that are connected will share one or more labels (with the exception of the null location []).

Naturally, a location’s connections do not include all possible sublocations or superlocations: this would be a redundant, unwieldy solution. Instead, an algorithm is used to choose connections, so that:

- No chosen sublocation is itself a sublocation to any other chosen sublocation;
- The chosen sublocations provide access to all locations that intersect with, or that are sublocations of, the current location;
- No chosen superlocation is itself a superlocation to any other chosen superlocation; and
- The chosen superlocations provide access to all locations that are superlocations of the current location.

The algorithm that will be used to compute these connection lists is still under development and testing.

**Implication links**

Many of the relationships between concepts can be expressed in the form of implication. For instance, in the user’s information repository, it may be the case that materials related to ASIS&T are always related to Research, and Diagrams are always Pictures. These implications can be used to aid organization. If the labels ASIS&T and UIST (and other conference/journal-related labels) are always accompanied by the label Research in location addresses, then the user can make use of this conceptual commonality while performing PIM tasks.

However, asking the user to manually enter both ASIS&T and Research every single time is unwieldy; furthermore, the user may not always remember to do this. Consider, for instance, the following locations:

- [Research, ASIS&T, LaTeX]
- [Research, ASIS&T, Diagrams]
- [ASIS&T, Notes]

Forgetting to add Research to the last location is a simple oversight. As a result, however, the user’s notes are not listed with the other materials as superlocations of [Research, ASIS&T]. This is another case of conceptual fragmentation, and is undesirable.

In order to address this problem, we focus on the fact that an implication is a general fact about a pair of concepts; therefore, the user should only have to specify it once. For this reason, the new PIM scheme allows the user to specify implication links. If the user specifies the link \( P \rightarrow Q \), then any location incorporating \( P \) will automatically incorporate \( Q \) as well. This allows the user to express and make use of implication relationships between concepts, without the burden of always having to remember to include the implied links.

**Manipulation**

This section discusses issues that arise when manipulating the abstract structure of an information repository.

**Accessing a location**

When dealing with the conventional, tree-based system, a folder must be created before it can be accessed. With the new scheme, however, the distinction between ‘creation’ and ‘access’ becomes more subtle.

If the user attempts to access a location containing a new label, the system will raise an exception: the user can either create the new label (allowing her to access the requested location), or abort the operation. However, if none of the labels are new, the user must be allowed to enter any combination of labels without explicitly having to ‘create’ the corresponding location. This is necessary in order to avoid entry point restriction. The user should be able to enter whatever labels she remembers (in whatever order she remembers them) as a location address, and then be given connection lists that allow her to navigate to the rest of the repository.

However, even if the user accesses the repository via an arbitrary location, this location may not be relevant to future PIM activities. In order to address this, we specify two types of location:

- **Persistent**: a persistent location is a permanent part of the information repository (barring future modification). A location is persistent if it satisfies one of the following conditions:
  - It has been declared ‘persistent’ by the user. (Note that this is done automatically if a new label is created while accessing a location, since it is reasonable to assume the user intends to use this location in the future.)
  - It contains data items. (This and the previous condition are referred to as intentional persistence.)
  - It is in the sublocation list of multiple other persistent locations. (This is referred to as structural persistence.)

Note that although the user can make any location intentionally persistent, either by declaring it so or by placing files in it, she can only revoke the intentional persistence of
an empty location. Furthermore, structural persistence cannot be revoked (although a structurally persistent location can be declared intentionally persistent).

- **Ephemeral:** an ephemeral location is one that is not persistent. It could be a non-persistent location that was directly accessed by the user; or, it could be a bridging location between a previously-accessed ephemeral location and the rest of the repository.

The user can access both persistent and ephemeral locations. When she accesses an ephemeral location, connection lists are dynamically generated for the location. (A caching policy for ephemeral locations could help speed up connection list generation.) However, in order to avoid cluttering the repository with locations that may not be relevant in the future, ephemeral locations are not formally remembered. They are not made a permanent part of the repository, and they are not listed in persistent locations’ connection lists. (Furthermore, if the user navigates from ephemeral location A to ephemeral location B, B will not possess a connection to A.) These changes are only made if an ephemeral location becomes persistent. (Note, however, that ephemeral locations should be preserved as part of navigational history.)

**Deleting a location**
Deleting a location is a cascading process: all of the location’s superlocations are also deleted. Note that deleting a location may cause one or more of the location’s sublocations to no longer fulfill the condition for structural persistence. This will render them ephemeral (and therefore forgotten), unless they were also made intentionally persistent beforehand.

**Creating an implication link**
When the user creates an implication link $A \rightarrow B$, this has the effect of adding B to any addresses that contain A but not B. (This may result in cascading changes to the information repository.) The user is declaring that A is only meaningful when accompanied by B, and the repository is being modified in order to be consistent with this new rule.

Consider, however, the case where $A + \text{Subset}$ and $B + A + \text{Subset}$ both contain items (or have both been made intentionally persistent). This produces a conflict: A + Subset has previously been declared meaningful in addition to B + A + Subset. This contradicts the rule being enforced by the implication link.

The user may cancel the creation of the implication link; or, she may attempt to *merge* the conflicting locations. This may cause further conflicts (due to identically named data items), which would require the items to be differentiated in some way (either by altering their names or by placing one set of items in a superlocation).

**Deleting an implication link**
When the user deletes a link $A \rightarrow B$, she is given the following options:

- **Soft deletion:** although the link is deleted, the structure of the repository is not changed. Although B is still present in any address incorporating A, it will not be added to new addresses incorporating A.
- **Hard deletion:** here, in addition to the link’s deletion, B is also deleted from any address incorporating A. Note, however, that if an address contains a label C that also implies B, then B will not be deleted. Furthermore, if B implies another label D, then the user will be asked whether she wants to remove D; and if $D \rightarrow E$, then the user will be asked whether she wants to remove E; and so on.

**BENEFITS OF THE NEW SCHEME**
The new PIM scheme was designed to address some of the issues present in the current, tree-based approach. In this section, we provide reasons why we consider our scheme to be a good start to resolving these issues.

- **Conceptual fragmentation:** if two or more locations share a common set of labels, then connections will be generated for those locations that quickly allow the user to travel from one to the other, ensuring that they remain closely connected, instead of being fragmented across the information repository.
- **Conceptual non-comprehensiveness:** because labels have universal scope, any location incorporating a certain label will have a quick path to all other locations that also incorporate that label – there is less chance that the user will miss a relevant location. Furthermore, implication links help the user to maintain a consistent addressing scheme (instead of forgetting to include Research along with ASIS&T in the address, for instance).

An autocomplete feature could help make sure that users reuse existing labels where appropriate, instead of creating new ones. This helps prevent fragmentation and non-comprehensiveness due to synonymy (eg. Literature search vs Literature review) (Quan et al., 2003).

- **Entry point restriction:** because the user can enter any combination of existing labels as a location address, she does not face restrictions when deciding where to begin her navigation.
- **Arbitrary ordering:** the user is not restricted to any particular label order when accessing a location.

**A PROTOTYPE BROWSER APPLICATION**
Although we have yet to implement a data storage system based on the new scheme, we have prepared a mockup interface for a prototype browsing application (figure 2). The address bar at the top allows users to enter addresses directly: while typing, an autocomplete feature will provide label suggestions, in order to avoid label synonymy. In the example location, Research is an implied label, which is why it is colored differently than the other labels in the address. When a label is entered that implies another label, the implied label is added to the bar automatically.

This brings up the issue of whether or not to display implied labels in the user interface. There are two perspectives here: on one hand, hiding implied links would reduce redundancy...
for a user who was familiar with the repository. Furthermore, it would improve address visibility if there happened to be a large number of implied labels in an address. On the other hand, a user who was not familiar with the repository (or the concepts used to construct it) might feel lost without the ability to see implied labels. Since both approaches have advantages and disadvantages, the browser application will allow users to choose whether or not to display implied labels.

The buttons to the left of the navigation panes allow the user to create and manage implication links. Here, we must address the issue of how to represent the set of implication links. Although it would be possible to simply display a list, visualizing the links as a directed graph might provide the user with useful insights about the structure of the information repository. However, for ease of management, the browser application will also include a list view. Furthermore, the graph view will have to be designed so that it can cope with a large number of links.

The three list views in the bottom right section of the interface are for navigation. The one on the left displays the current location’s connected sublocations, while the one on the right displays connected superlocations. The center list displays the data items stored in the current location. In most visualizations of the tree-based approach, subfolders are displayed as members of their parent folder, along with the files located at that folder. For our interface, however, we have chosen to display files and superlocations separately, in order to emphasize that superlocations are not contained by a given location – a location may appear in multiple lists of connected superlocations.

This interface visualizes one location at a time, along with its connected sublocations and superlocations. However, developing an effective way to visualize larger portions of the information repository is an important goal. One of the benefits of the tree-based approach is that trees are relatively easy to visualize; the new scheme, on the other hand, has a more complex underlying structure that will require more effort to visualize in an easily-comprehensible way.

**FUTURE RESEARCH**

We plan to implement a data storage system based on our new scheme, in addition to a browser application for accessing and manipulating the data stored in the system. The algorithms that will be required for implementation are still under development. We will attempt to apply existing graph processing algorithms to our abstract structure, in order to leverage existing work and avoid reinventing the wheel. It will also be necessary to study the graph structure formed by implication links, in order to identify and resolve potential problems.

For initial development and testing, we feel it makes most sense to implement the data storage system using existing database software, instead of trying to implement a file system from scratch. Implementing the browser application will require us to pick a target platform, in addition to a UI toolkit. Furthermore, for testing purposes, it will be necessary for us to implement a visually similar application that provides access to a conventional, tree-based information repository.

There are many possible methods for interacting with the new PIM scheme that we have not studied yet. For instance, we believe there is a rich set of operations users can perform directly on labels, such as splitting labels (converting A to A and B), renaming them (converting A to B), etc. Identifying these operations, studying their effects on the abstract structure of an information repository, and presenting them to users in a clear, intuitive fashion, constitute a significant topic for further research. There are also basic file operations (such as moving and copying locations) that we have not translated to the new scheme yet – this will also be a substantial research effort.

In order to determine the usefulness of the new scheme, we will need to conduct user tests, comparing the new scheme with the tree-based approach. Our initial studies will be short-term tests that have subjects attempt common PIM tasks using one of the two schemes, or both (depending on the study design). However, a longitudinal study would also be valuable.
CONCLUSION
Despite being well-established and universally deployed, the tree-based approach to personal information management has several problems, some of which we have described in detail. We have proposed a new scheme for PIM that uses shared labels and implication links to create an information repository where locations that are related conceptually are closely connected. We have explained how the new scheme might address the problems present in the tree-based approach, and we have presented a mockup of a browser application for accessing and manipulating repositories that make use of the new scheme. Our next step will be to implement a data storage system based on the new scheme, and to conduct user studies in order to quantitatively compare the new scheme with the existing tree-based approach.

REFERENCES