# Creating a Comparative Environment for PIM Evaluation

Karl Voit
Institute for Software Technology (IST)
Graz University of Technology
Karl.Voit@IST.TUGraz.at

Keith Andrews
Institute for Information Systems and Computer Media (IICM)
Graz University of Technology
kandrews@iicm.edu

Wolfgang Slany
Institute for Software Technology (IST)
Graz University of Technology
Wolfgang.Slany@tugraz.at

## ABSTRACT

A common form of comparative evaluation in current research is to run a formal experiment with a number of test users and software implementations of two or more underlying research approaches. When the underlying methods are, in fact, the subject of the comparison, it is essential for fair comparison that their software implementations and interface usability be of equivalent quality. Previous work on evaluating information visualisation techniques can inform such evaluations in the area of human-computer information retrieval. A research framework called *tagstore* supports experiments on different styles of tagging interfaces. It can also be used for experiments involving the storing and re-finding of personal files using either folder hierarchies or tagging. The tagstore framework provides a basis for comparable results, flexible control of appearance and functionality, and multi-platform availability.

## Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: User Interfaces

## Keywords

personal information management, information retrieval systems, file storage, hierarchical structures, dynamic structures, tagging, faceted search

## 1. INTRODUCTION

A commonly used evaluation method in areas like Personal Information Management (PIM), Information Retrieval (IR), and Human-Computer Interaction (HCI) research is comparative evaluation by conducting a formal user study. Typically, test users are asked to accomplish tasks using two or more software interfaces. Objective and subjective data are collected and analysed. Researchers often seek to draw conclusions regarding the underlying techniques, methods, or algorithms, which lie beneath the software interface.

Unfortunately, in some studies, the software used for comparison is often written by different groups, to different standards, and using differing developmental resources. Hence, the quality of the software implementation might well influence the outcome of the experiment and lead to conclusions which are not valid.

Figure 1 illustrates the fundamental problem: the interface layer is necessary for any software implementation of a method. In formal experiments, positive or negative effects due to differences in the implementation layer are often neglected.

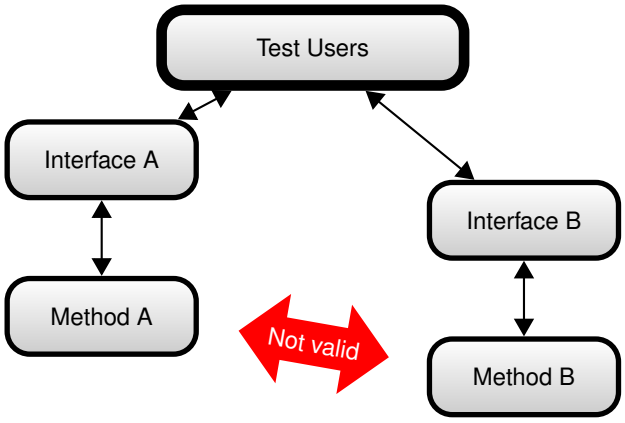
## 2. RELATED WORK

Kobsa [10] compared five different information visualisation techniques for navigating in hierarchies with Windows Explorer. The software for each technique was provided by the originating research group. The implementations of the methods vary in quality, usability, feature set, target applications, degree of maturity, and in the case of Windows Explorer clearly in user familiarity. Hence, the findings are highly dependent on the implementation differences and not solely in the methods used to visualize the data. As the author concedes: "the extreme outliers in the Tree Viewer and BeamTrees data have mostly been caused by a lack of functionality beyond the pure visualization." When using software tools from completely different sources, almost any result is dependent on the implementation layer to a large extent.

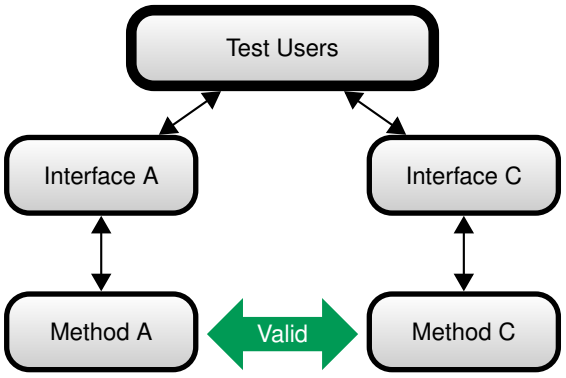
The comparative study described by Andrews and Kasanicka [4] used four different information visualisation techniques. However, the various techniques were deliberately re-implemented by the same research group within a single framework, in an attempt to control for the quality of implementation.

Munzner [14] explicitly distinguishes four levels of evaluation or validation in the context of information visualisation systems: 1) domain problem characterisation (addressing the wrong issue), 2) data abstraction design (choosing the wrong abstraction), 3) interaction design (poor usability) and 4) algorithm design (inefficient implementation). It is argued that each level, including the usability and implementation levels, must be validated in and of itself, using the appropriate techniques at each level. This question of when to use which evaluation technique was also previously addressed by Andrews [3, 2].

The previous examples were from the field of information visualisation. Turning to the fields of PIM and HCIR, Civan et al. [7] looked into the question as to whether browsing (hierarchical structures) or tagging is more efficient for storing and re-finding information. An experiment was constructed using two different web-based email systems: Hotmail and Gmail. The general assumption was that the main difference between those two systems are that Hotmail uses folders and Gmail uses tags to organise emails.




(a) Test users taking part in a formal experiment are typically asked to complete tasks on a number of alternative interfaces. Their relative performance is then statistically compared. However, in general, it is risky to draw conclusions about underlying methods, without first ensuring that their implementations are of equivalent quality.




(b) When indirectly testing two methods through the same or equivalent interfaces, it is more reasonable to draw conclusions about the underlying methods. The implementations have to be of comparable quality and the featureset should differ only in things that are subject of the experiment.

**Figure 1: When evaluating two methods using formal experiments, the experimental design has to provide for comparable interfaces.**

"Most differences between these systems" were considered "minor and could be overlooked by participants for the purpose of [the] study". Besides the fact that Hotmail and Gmail are two different interfaces hardly comparable, the tested version of Hotmail did not have a representation of folders that meets common definitions of "folders": there was no possibility to create sub-folders and every given folder was visible all the time. Navigation is unnecessary, since Hotmail's "implementation of folders" is closer to "tagging with only one tag per email" than using hierarchical folder structures. Hence, the results of this study are not applicable at least for systems that provide navigational folder structures.

Another study which tried to answer a similar research question was described in [15]. The second study described in that paper was done using a self-implemented software, which differed only in one single feature: storing and re-finding photographs using either tags or folders. This study controls for implementation variability and permits much better generalisation of the results to the underlying methods: folders versus tags. Unfortunately, there were also some issues in the study which limit more general application of the results: items were digital photographs only, sub-folders were visible all the time without any possibility to collapse sub-trees, and the tagging process lacked commonly provided features, such as tag completion and tag recommendations.

Ideally, software being used for a comparative study should be designed so as to eliminate any differences in interface or implementation and hence facilitate the application of any results to the underlying methods, as shown in Figure 1(b).

## 3. TAGSTORE: A RESEARCH FRAMEWORK FOR TAGGING INTERFACES

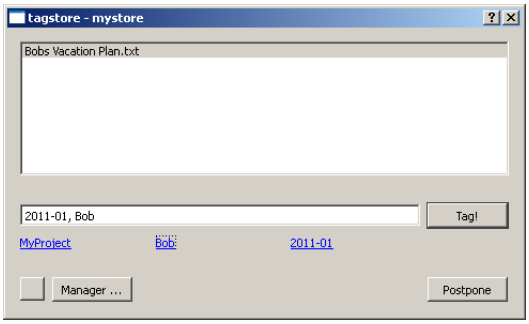
To investigate the best way to design a tagging interface for personal information management, a research framework called *tagstore* was built [18, 20]. Previous work discussed requirements for user acceptance of PIM systems [19]. All of these requirements are implemented in tagstore. In relation to user studies, tagstore focuses on providing the best possible usability, compatibility with the current software environments, multi-platform availability, and test users needing no advanced computer knowledge.

Inside the framework, tags are mapped to the file system as follows: for any given file or folder (item) added to tagstore, the system automatically generates navigational folder and link structures called *TagTrees*. Within a TagTree, every permutation of tags assigned to a particular item is stored as a navigational folder path. Each folder along a path represents a specific tag for one or more items and contains symbolic links to the original items. The items themselves are stored in a separate central storage folder. Users navigate a TagTree by selecting one tag at each step along the way. All permutations of tags are provided, so users may select tags in any desired order. Item renaming or deletion automatically results in updates of the TagTrees correspondingly.

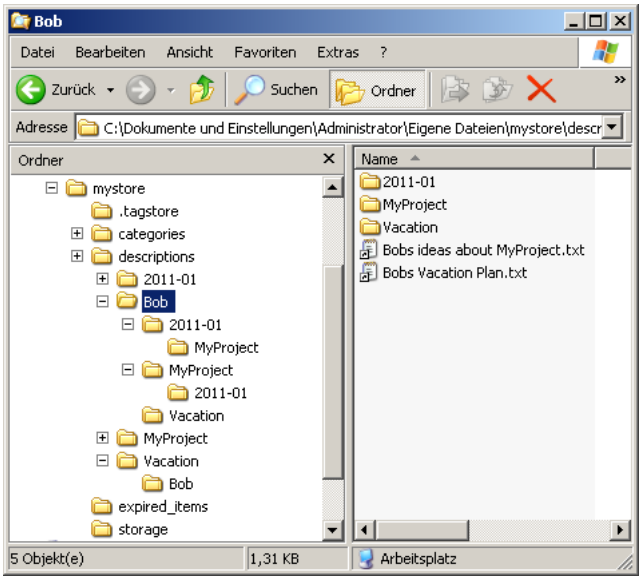
TagTrees provide associative navigation paths. Therefore the domain of tagstore is navigation and not search. Navigation is the preferred method of information re-finding as studies like [1, 17, 5, 6] show. Using navigation, users do not have to formulate queries and are able to re-find files they already forgot. Desktop search engines are an additional method to locate (known) files. This framework covers storing and re-finding of information provided within files or folders only. But the general method can be adopted to other systems as well.

If a file called `Ideas.txt` is stored in tagstore with a number of $n = 3$ tags – "Project", "important", and "Bob" – then the six generated navigation paths comprising the TagTree are: `Project/important/Bob`, `Project/Bob/important`, `important/Project/Bob`, `important/Bob/Project`, `Bob/important/Project`, and `Bob/Project/important` respectively. In each (sub-)folder, the user finds a link to `Ideas.txt` resulting in $n \times n!$ additional links to the item. The deeper the user navigates, the fewer links to other items are visible, and the more detailed the implicit "query" becomes.

Similar ideas of mapping navigational structure elements into the file system can be found in Mohan et al. [13] and Gifford et al. [8].




(a) The tagstore dialog window. The user is assigning tags to the file `Bobs Vacation Plan.txt`. At the top is the list of untagged files in the current store. At the bottom is the tagline showing two entered tags and beneath it are three tag suggestions in blue.




(b) Windows Explorer showing the automatically generated TagTree navigational structures: folder structures in the file system are generated to correspond to every permutation of all tags assigned to an item. These can then be navigated using standard tools like Windows Explorer.

**Figure 2: The tagstore tagging process with one tagline and a free vocabulary.**

The TagTrees method is neutral to semantics and can be implemented without any relational database management system or special in-between file system layer.

To store an item in a TagTree structure, the item has to be copied, moved or saved to the central `storage` folder. This folder should be known to and easily reachable by the user. An application-independent tagging dialog automatically appears and allows the user to enter appropriate tags for the item.

This approach has many advantages. Most of all – at least for re-finding – the user has access to additional functionality without an additional interface. The TagTree structure can be navigated with any current file management application, in every file Open or Save dialog window, and with both GUI-based applications and command-line tools. Any given backup process is unaffected, users handle their files as usual and retain confidence in the integrity of their data.

Figure 2(a) shows the tagging dialog. This dialog appears whenever an item is placed in the central `storage` folder. Entering tags is designed to be as easy and as fast as possible. A recommender system proposes tags which might be helpful to the user in the current context. Autocompletion is provided for previously used tags. Multiple items with the same tags can be tagged in one step, and previously entered tags are the default tags in the next tagging dialog. When the user finishes tagging, TagTrees related to the item are created. Figure 2(b) shows a partly expanded view of TagTrees in Windows Explorer.

The current implementation of tagstore has some technical limitations: the number of inodes[1] is limited. Due to the exponential growth of folders and links inside a TagTree structure, many items with many tags per item result in a large number of inodes being used in the current hard disk partition. This results in a reasonable upper bound of a few thousand items per tagstore[2]. For research purposes, this limitation is acceptable, since a few thousand items are sufficient for most experimental situations.

For performance reasons, the number of tags per item is limited to six or seven, depending on the hardware and operating system[3]. As other studies have shown, users very seldom assign more than five tags per item [9, 15]. In longer term field tests of tagstore, users were hardly ever shown the warning message related to this limit.

## 4. USING TAGSTORE FOR RESEARCH

For researchers conducting an experiment, tagstore offers a wide range of possibilities to change appearance and behavior, it can be used in multi-language (currently, English and German) experiments, and its open source licence (GNU GPL v3) allows it to be freely enhanced or expanded. All data and configuration settings are stored in easily parsable plain text files. Finally, tagstore is available for all three major operating systems: Microsoft Windows (Vista or newer), Mac OS X, and Debian GNU/Linux based distributions like Ubuntu.

Körner et al. [11] introduces the concept of dividing users of tags into *describers* and *categorisers*. tagstore allows a second tagline to be configured. Thus, one tagline can be used for descriptive tags and the second tagline for category tags. With two distinct taglines,

[1]The smallest information chunks file systems can handle.

[2]Users are able to create and use several tagstores in parallel: for example one store each for work, photographs, and miscellany.

[3]Solid State Disks (SSD) are much more faster than standard hard drives; symbolic links (GNU/Linux and Mac OS X) are faster than the shortcuts used by Microsoft Windows.

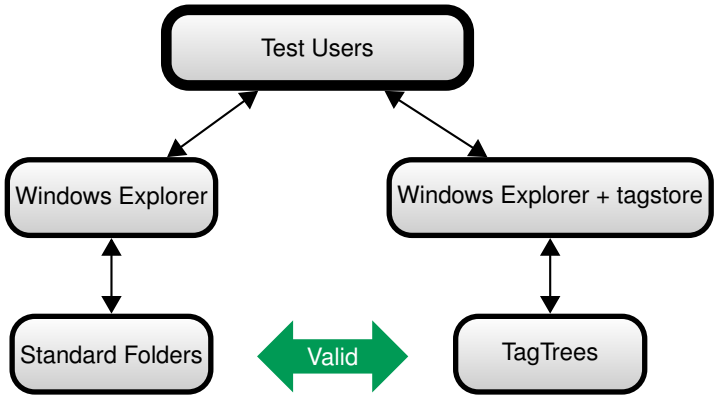



**Figure 3: For experiments where tagging versus filing into traditional folder structures is the subject, tagstore – as an additional implementation extending the traditional Windows Explorer interface – introduces only one dialog window that handles the tag assignment. The underlying method of re-finding items does not differ (in terms of interface) at all. TagTrees are accessed by Windows Explorer (or similar tools) and provide associative navigation to items rather than remembering storage paths in a strict hierarchy.**

a comparative study can be run to assess their relative use.

tagstore supports the use of a controlled vocabulary (CV) for the set of tags. This is called "My Tags" and can be maintained in the tagstore Manager. The use of a CV is a common method to avoid problems with homonyms, synonyms, and plurals. Again, comparative studies can be formulated to compare the use or non-use of a controlled vocabulary.

Date stamps can be configured as default tags in tagstore (added automatically). Furthermore, it is possible to explicitly define expiry dates for each item. As Mayer-Schönberger [12] suggests, expiry dates allow for promising PIM research and could help in alleviating information overload.

Finally, researchers might use tagstore to conduct studies comparing the use of folder structures to the use of tagging. Although the assignment of tags in tagstore involves the use of a custom dialog window, navigation through TagTree structures is accomplished with exactly the same Explorer-like tree browser interface used to navigate and maintain folder hierarchies. Hence, for this kind of experiment, tagstore provides a "minimally invasive" environment using the very same, well-known hierarchy browsing interface. This ensures that results of experiments are being able to compared to classical re-finding tasks using navigating in strict hierarchies of folders as Figure 3 outlines.

## 5. CONCLUDING REMARKS

Comparing methods using software implementations is not easy to do. Implementation differences, independent of the underlying methods, can be the reason for incomparable results. The tagstore research framework provides a rich set of equivalently implemented features for researchers to run experiments around tagging interfaces. It can also be used to compare tagging to browsing in folder hierarchies.

Possible future enhancements include: alternative structures for TagTrees such as those proposed by Solskinnsbakk and Gulla [16], the introduction of semantic information, different styles of tagging window, alternatives to TagTree navigation in an browser, cross-platform studies, and so on.

Two formal experiments were conducted using the tagstore framework. Preliminary results show a more diverse picture than previous studies in this field. The detailed findings from these experiments will be published in forthcoming papers. Complete data sets will also be published in the spirit of the Open Science movement[4]. Additionally, a field test is being developed to examine the long term effects of using a PIM tool which provides associative navigation. In the field test, test users will be able to use tagstore over an extended period of time with their own data and their own use cases.

## 6. REFERENCES

[1] C. Alvarado, J. Teevan, M. S. Ackerman, and D. Karger. *Surviving the information explosion: How people find their electronic information*. AI Memo AIM-2003-006, MIT AI Laboratory, Department of Computer Science, 2003. http://hdl.handle.net/1721.1/6713.

[2] K. Andrews. *Evaluating Information Visualisations*. In *Proc. AVI 2006 Workshop on Beyond time and errors: novel evaLuation methods for Information Visualization (BELIV'06)*, pages 1–5. ACM Press, May 2006. 1595935622. doi:10.1145/1168149.1168151.

[3] K. Andrews. *Evaluation Comes in Many Guises*. CHI 2008 Workshop on Beyond time and errors: novel evaLuation methods for Information Visualization (BELIV'08), April 2008. http://www.dis.uniroma1.it/beliv08/pospap/andrews.pdf.

[4] K. Andrews and J. Kasanicka. *A Comparative Study of Four Hierarchy Browsers using the Hierarchical Visualisation Testing Environment (HVTE)*. In *Proc. 11th International Conference on Information Visualisation (IV'07)*, pages 81–86. IEEE Computer Society Press, July 2007. doi:10.1109/IV.2007.8.

[5] D. Barreau. *The Persistence of Behavior and Form in the Organization of Personal Information*. *Journal of the American Society for Information Science and Technology*, 59(2):307–317, January 2008. ISSN 1532-2882. doi:10.1002/asi.20752.

[6] O. Bergman, R. Beyth-Marom, R. Nachmias, N. Gradovitch,

[4] http://en.wikipedia.org/wiki/Open_science

and S. Whittaker. *Improved Search Engines and Navigation Preference in Personal Information Management*. *Transactions on Information Systems*, 26(4):1–24, September 2008. ISSN 1046-8188. doi:10.1145/1402256.1402259.

[7] A. Civan, W. Jones, P. Klasnja, and H. Bruce. *Better to Organize Personal Information by Folders or by Tags?: The Devil is in the Details*. *Proceedings of the American Society for Information Science and Technology*, 45(1):1–13, 2008. ISSN 00447870. doi:10.1002/meet.2008.1450450214.

[8] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. James W. O'Toole. *Semantic File Systems*. In *Proc. 13th ACM Symposium on Operating Systems Principles (SOSP 1991)*, pages 16–25. ACM, October 1991. doi:10.1145/121132.121138. http://cgs.csail.mit.edu/history/publications/Papers/sfs.ps.

[9] J. L. Hsieh, C. H. Chen, I. W. Lin, , and C. T. Sun. *A Web-based Tagging Tool for Organizing Personal Documents on PCs*. In *International Conference of Computer-Human Interaction 2008 (CHI2008). Florence, Italy.* April 2008. http://works.bepress.com/lucemia/18/.

[10] A. Kobsa. *User Experiments with Tree Visualization Systems*. In *Proc. IEEE Symposium on Information Visualization (InfoVis 2004)*, pages 9–16. Austin, Texas, USA, October 2004. doi:10.1109/INFVIS.2004.70. http://www.ics.uci.edu/~kobsa/papers/2004-InfoVis-kobsa.pdf.

[11] C. Körner, D. Benz, A. Hotho, M. Strohmaier, and G. Stumme. *Stop thinking, start tagging: tag semantics emerge from collaborative verbosity*. In *Proceedings of the 19th international conference on World wide web*, pages 521–530. ACM, New York, NY, USA, 2010. 1605587990. doi:10.1145/1772690.1772744.

[12] V. Mayer-Schönberger. *Delete: The Virtue of Forgetting in the Digital Age*. Princeton University Press, October 2009. 0691138613.

[13] P. Mohan, V. S. Raghuraman, and A. Siromoney. *Semantic File Retrieval in File Systems Using Virtual Directories*. Poster Session of the 13th Annual IEEE International Conference on High Performance Computing (HiPC), Bangalore, India, December 2006.

[14] T. Munzner. *A Nested Process Model for Visualization Design and Validation*. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, November 2009. ISSN 1077-2626. doi:10.1109/TVCG.2009.111. http://cs.ubc.ca/labs/imager/tr/2009/NestedModel/NestedModel.pdf.

[15] R. Pak, S. Pautz, and R. Iden. *Information Organization and Retrieval: A Comparison of Taxonomical and Tagging Systems*. *Cognitive Technology*, 12(1):31–44, 2007. http://business.clemson.edu/Catlab/pubs/pak-pautz-iden-2007.pdf.

[16] G. Solskinnsbakk and J. Gulla. *A Hybrid Approach to Constructing Tag Hierarchies*. In R. Meersman, T. Dillon, and P. Herrero, editors, *On the Move to Meaningful Internet Systems, OTM 2010*, volume 6427 of *Lecture Notes in Computer Science*, pages 975–982. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-16949-6_22.

[17] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. *The Perfect Search Engine is not enough: a Study of Orienteering Behavior in Directed Search*. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 415–422. ACM, New York, NY, USA, 2004. 1-58113-702-8. doi:10.1145/985692.985745. http://people.csail.mit.edu/teevan/work/publications/papers/chi04.pdf.

[18] K. Voit. *tagstore — Project home page*, November 2011. http://tagstore.org/.

[19] K. Voit, K. Andrews, and W. Slany. *Why Personal Information Management (PIM) Technologies Are Not Widespread*. In *PIM09 ASIS&T 2009 Workshop, Vancouver, BC, Canada*, pages 60–64. 2009. http://pimworkshop.org/2009/index.php?page=acceptedpapers.

[20] K. Voit, K. Andrews, and W. Slany. *TagTree: Storing and Re-finding Files Using Tags*. In *Proc. 7th Conference of the Austrian Computer Society Workgroup: Human-Computer Interaction (Usab 2011)*, volume 7058 of *LNCS*, pages 471–481. Springer, November 2011. 3642253636. doi:10.1007/978-3-642-25364-5_33.