

Karl Voit

TagTrees: Improving Personal Information Management Using Associative Navigation

Dissertation

Graz University of Technology

Institute for Softwaretechnology

Head: Univ.-Prof. Dipl-Ing. Dr. techn. Wolfgang Slany

Supervisor: Univ.-Prof. Dipl-Ing. Dr. techn. Wolfgang Slany

Co-Supervisor: Ao. Univ.-Prof. Dipl-Ing. Dr. techn. Keith Andrews

Graz, November 1, 2012

This document was written with [GNU Emacs](#), is set in Palatino, compiled with [pdfL^AT_EX2_ε](#) and [Biber](#).

All figures, which do not refer to an external source, are made by the author using [Ipe](#), the extensible drawing editor.

The L^AT_EX template, which was made by the author, is based on [KOMA script](#) and can be found online: <https://github.com/novoid/LaTeX-KOMA-template>.

Paper: Bio Top 3 *extra* (80 g) (Mondi Neusiedler).

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material, which has been quoted, either literally or by content from the used sources.

Graz, _____
Date Signature

Eidesstattliche Erklärung¹

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am _____
Datum Unterschrift

1. Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Acknowledgments

At this point I would like to thank the wonderful group of students who were part of this research project over the last four years: Georg Achleitner, Johannes Anderwald, Gerulf Binder, Laurens De Vocht, Andrea Denger, Christoph Friedl, Daniel Fussenegger, Annemarie Harzl, Vesna Krnjic, Manfred Oberlerchner, Michael Pirrer, Florian Praxmair, Georg Schober, Alexander Skiba, Barbara Stadlhofer, Matija Striga, Josef Wachtler, Mario Wiedner, Armin Wieser, and Wolfgang Wintersteller. With your impressive passion and great contributions you made all this possible. It was a pleasure working with you and you taught me many things!

My supervisors, Wolfgang Slany and Keith Andrews, provided me guidance and help when I was unsure or helpless. I enjoyed wonderful discussions and your opinions were very valuable to me.

Furthermore, I would like to thank my colleagues and predecessors in [Personal Information Management \(PIM\)](#) research. Your work has inspired me so many times and I still love to read your books and papers. Doing research in this topic is very rewarding to me and enormously relevant to countless people out there!

My beloved parents, Elisabeth and Karl Voit, always encouraged me on this journey called life. Valuing education they made it possible for me to learn interesting things and become the person I am today for which I am very grateful.

Last, but surely not least, I would like to thank Petra Heidenkummer who also contributed to my research work so many times with discussions, opinions and encouraging support when I needed it. I love you, my dear!

Karl Voit

Contents

Abstract	1
1 Introduction	3
2 Research Background	9
2.1 PIM Strategies	10
2.1.1 As We May Think	10
2.1.2 How People Organize Their Desk	12
2.1.3 The Psychology of PIM	13
2.1.4 Cross-Tool Behavior and Longitudinal Data	17
2.2 PIM Tools	21
2.2.1 Sketchpad	21
2.2.2 NLS	22
2.2.3 Lifestreams	23
2.2.4 TimeScape	25
2.2.5 Remembrance Agent	26
2.2.6 Presto	28
2.2.7 Haystack	29
2.2.8 WorkspaceMirror	32
2.2.9 Stuff I've Seen	33
2.2.10 Phlat	37
2.2.11 Attribute Browser	41
2.2.12 MyLifeBits	42
2.2.13 Semantic File System	45
2.2.14 SemFS	46
2.2.15 TagFS	47
2.2.16 hFAD	47
2.2.17 Feldspar	49
2.2.18 Personal Project Planner, Planz	51

2.2.19	LiFiDeA and Associative Navigation	55
2.2.20	Faceted Search, Faceted Navigation	59
3	Challenges	63
3.1	Changing Environments	63
3.1.1	Increasing Number of Files	64
3.1.2	Non-Textual Content	66
3.1.3	Scattered Storage Locations	67
3.2	Users	68
3.2.1	Time Spent While Retrieving Information	68
3.2.2	The Need for Diverse Interfaces	69
3.2.3	Teaching and Learning PIM	70
3.3	Tools and Development	73
3.3.1	Missing Availability of Advanced Solutions	73
3.3.2	Degree of Maturity of (Tagging) Interfaces	76
3.4	Research Topics	76
3.4.1	Desktop Metaphor as a Limiting Factor	77
3.4.2	Neglecting the Importance of Navigation	82
3.4.3	Local File Management Not Elaborate Enough	84
3.5	Research Methods	87
3.5.1	Observation is Not Always Enough	88
3.5.2	Long-Term Versus Short-Term	89
3.5.3	Using Only Iterative Processes	90
3.6	Research Questions	90
3.6.1	How Can Navigational Re-finding Be Improved?	90
3.6.2	How to Organize Information In a Better Way?	92
3.6.3	How to Evaluate Such a Method?	92
4	The TagTrees Method	93
4.1	Motivation	93
4.2	TagTrees	94
4.3	TagTrees and File Management	96
4.4	TagTrees Addressing Previous Research	99
4.5	The Effort of the Storage Process	101
5	tagstore Implementation	107
5.1	Motivation	108

Contents

5.2	Software Modules	110
5.3	tagstore Dialog	112
5.3.1	Tag Recommendations	114
5.4	tagstore Manager	115
5.4.1	My Tags	116
5.4.2	Datestamps	116
5.4.3	Expiry Date	118
5.4.4	Re-Tagging	120
5.4.5	Rename Tags	120
5.4.6	Store Management	121
5.4.7	Sync Settings	122
5.5	Internal Storage Structure	123
5.6	Workflows	127
5.6.1	Installation	128
5.6.2	Updating the Installation	131
5.6.3	Adding Items	131
5.6.4	Retrieving Items	132
5.6.5	Renaming Items	133
5.6.6	Deleting Items	133
5.6.7	Getting Help	134
5.7	Best Practices	134
5.8	Technical Limitations	135
5.8.1	Resources	135
5.8.2	Speed	136
5.8.3	Workarounds and Solutions	138
5.8.4	No Adding, Deleting, or Renaming in TagTrees	139
5.9	Comparing tagstore to Other Solutions	139
6	Evaluation	141
6.1	Strategy	142
6.2	Informal Feedback from Long-Term Users	143
6.3	Formal Experiment 1	145
6.3.1	Methodology	145
6.3.2	Results	161
6.3.3	Discussion	171
6.4	Formal Experiment 2	172
6.4.1	Methodology	173

Contents

6.4.2	Results	186
6.4.3	Discussion	202
7	Summary and Outlook	205
	Bibliography	213
	Glossary	231
	Index	241
	List of Figures	245
	List of Tables	249

Abstract

This dissertation gives an overview of research related to [Personal Information Management \(PIM\)](#). After discussing some important challenges, a new method for managing local files is described: *TagTrees*. Navigational structures for re-finding are automatically generated in the file system after the user has assigned tags to items (files or folders). These TagTrees provide for associative navigation, because each item is reachable via a large set of navigational paths, in the file system hierarchy.

The TagTrees method was implemented in a research software called *tagstore*. This implementation provides multiple possible configurations, making tagstore a research framework for analyzing many different kinds of tagging processes.

Besides long-term usage patterns, two formal experiments were conducted to evaluate tagstore. Comparative studies showed that tagstore performed similarly to the traditional folder hierarchy method. Subjective user feedback and acceptance was very positive. Having a comparable performance for filing, the additional possibilities for re-finding items combined with very high user acceptance makes TagTrees and tagstore a valuable contribution to [PIM](#) research and end user software.

In the spirit of open science and the reproducibility of research results, the tagstore framework, experimental materials, study results and evaluation tools are all publicly available under an open license.

Life is very short, and there's no time
For fussing and fighting, my friend
I have always thought that it's a crime
So I will ask you once again
Try to see it my way
Only time will tell if I am right or I am wrong

We Can Work It Out
The Beatles

1 Introduction

For the last twenty thousand years, mankind learned to adapt the environment to its needs. We developed tools and ways to use nature's resources like fire or water for our own purposes. Beginning with early cave painting and storytelling, we began to record and distribute information. The invention of writing reduced variation of content due to subjective interpretation. It furthermore allowed propagation and archiving of information independently of individual humans.

The concept of printing with woodblocks (before 220 AD) and movable types (around 1040 AD) enabled mass production and mass distribution in an enduring way. Having to manage a growing number of books, librarians began to adapt their methods for organizing scrolls and manuscripts to these new requirements. Following the idea of Aristotle, the world of knowledge was structured in a hierarchy of categories, a taxonomy.

With the development of modern information technology, in 1969 Multics contained a hierarchical [file system](#) comparable to the systems we are using now (see Corbató and Vyssotsky, 1965). This file system was the first to allow [directories](#) with an arbitrary number of sub-directories with long directory names, each holding [files](#) and further sub-directories.

Alan Kay developed the “[desktop metaphor](#)” for computers. It was introduced in 1970 at Xerox PARC for the Xerox Alto (Thacker et al. 1979),

1 Introduction

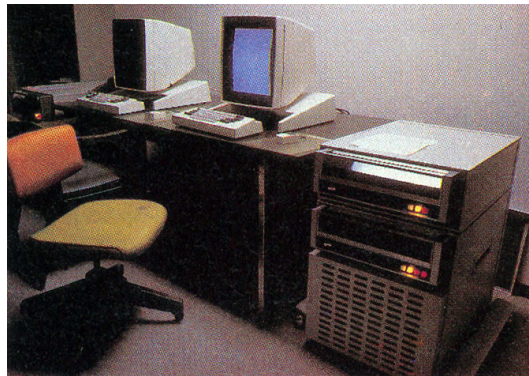


Figure 1.1: Xerox Alto. “Each Alto processor is made of medium- and small-scale TTL integrated circuits, and is mounted in a rack beneath two 3-megabyte hard-disk drives. Note that the video displays are taller than they are wide and are similar to a page of paper, rather than standard television screen.”

[Source: <http://www.digibarn.com/collections/software/alto/index.html> – retrieved on 2012-07-21]

Figure 1.1). The **Graphical User Interface** (GUI) took advantage of a user’s knowledge of real world things to describe non-physical interaction features and data. For example, physical folders were mapped onto the technical concept of directories in the file system. These ingenious ideas allowed people to use computers, without knowing about the internals of a computer at all. The directory metaphor made it possible to store information in a similar way to the methods that were used to file physical paper files and folders in drawers, stacks, and filing cabinets.

On the one hand, the introduction of the desktop metaphor was a very clever way of allowing early computer users to apply knowledge and concepts they already knew from the physical world. This enabled them to use a complicated machine more easily. On the other hand, the desktop metaphor also transferred one specific limitation of the physical world into the virtual world: one thing has *only one* specific location in storage.

For decades, the number of information entities per user has been increasing. Even worse, the number of information sources and storage locations has increased as well. Cloud services offer a variety of web services to organize and re-find information entities of various kinds. Their user interfaces

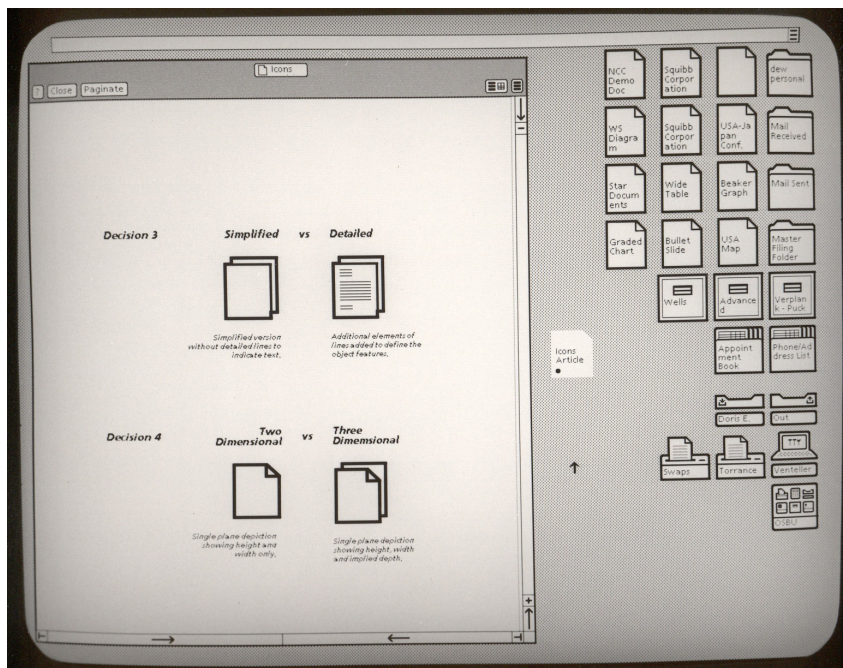


Figure 1.2: The Xerox Star 8010 graphical user interface in year 1981. The Xerox Star (Johnson et al., 1989) was a direct successor of the Xerox Alto (Thacker et al., 1979). The desktop metaphor is already in an advanced state. There is a window with a title, window manipulation buttons and scroll bars. The icons on the desktop represent documents, folders, network drives, email boxes and printers.

[Source: <http://www.digibarn.com/collections/screenshots/xerox-star-8010/index.html> – retrieved on 2012-07-21]

are constantly being optimized to handle the increasing amount of data. However, there is still a huge amount of information on local hard drives, organized in the traditional way: filing in a [folder hierarchy](#). Organizing up to a few hundred files within a hierarchy of [folders](#) does not seem to be a problem. Having hundreds of thousands of files within the same hierarchy is a problem.

Users often prefer special-purpose software to manage files of different formats. They use music management products to manage music files with metadata related to music. This way, they are able to derive different kinds of views according to album, artist, year of publication, song duration, and

1 Introduction

so forth. Users install digital photo management software to handle their digital photo library, so that they are able to derive different kinds of views according to events, people in the photographs, topics, colors and so forth. These special-purpose information management products demonstrate the limited capabilities of our main information management software layer: the file system.

Since the seventies, many workarounds were developed to allow more flexible structures: for example [hard links](#), [symbolic links](#), and [shortcuts](#). Disregarding the fact that users seldom use such file system features, they do not even provide an effective solution to the underlying problem: [broken links](#), multiple copies of the same file, and frustrated users are the result. Users have adapted to the limitations of a [strict hierarchy](#) of folders. They keep implicit cognitive models of “their system” in their minds. When a file relates to multiple different destination folders, those cognitive models decide (mostly unconsciously) which folder “wins”. There are many problems related to this. Not only that these cognitive models change over time for a single user, the problem becomes even worse when multiple users work with a shared hierarchy of folders. Nevertheless, people have been basically using the very same unsatisfactory method to manage more and more files (and folders) for the last sixty years without any fundamental improvement.

[Personal Information Management \(PIM\)](#) research over the last decades developed a number of methods, prototypes and products. Chapter [2](#) describes relevant research work and prototype interfaces.

This thesis focuses on [navigation](#) within a [file system](#), excluding [search](#) methods. Although [desktop search](#) engines are a useful alternative retrieval method for many users, several studies have shown that the majority of users prefer navigation over search for access to local files and folders. Chapter [3](#) discusses some challenges related to PIM and motivates the basic principles described in this thesis. Several areas where PIM research has great potential I described.

In Chapter [4](#), a new method for organizing and navigating to local files for a single user is introduced: instead of filing files or folders into a hierarchy of folders, the user tags them. The system uses these tags to automatically derive a navigational hierarchy, called *TagTrees*. The structure of TagTrees is

designed to allow [associative navigation](#) in contrast to having to remember storage locations. A user has many different possible navigational paths to the same information entity.

The TagTrees method was implemented in research software called *tagstore* whose technical details and features are described in detail in [Chapter 5](#). Best practices and technical limitations of the current implementation are elaborated on as well.

Long-term user tests and two formal experiments were conducted to evaluate the tagstore implementation. [Chapter 6](#) describes the research insight which was gained while comparing tagstore to the common folder filing method. Objective performance measures showed a mixed picture, but subjective participant feedback clearly favored tagstore.

[Chapter 7](#) summarizes this thesis and presents some ideas for possible future work and development.

The word is about, there's something evolving,
Whatever may come, the world keeps revolving
They say the next big thing is here,
That the revolution's near,
But to me it seems quite clear
That it's all just a little bit of history repeating

History Repeating
Propellerheads (Feat. Miss Shirley Bassey)

2 Research Background

Research contributions in the field of PIM are most often accomplished by confronting test persons with a tool, method, or questions or by observing test persons. Figure 2.1 gives a short overview of the influence vectors and the possibilities of gaining objective or subjective feedback. The things that are changed during an experiment are called *independent variables* and the measures which we observed or derived are called *dependent variables* (McGrath, 1995). Early research contributions, especially, consisted mostly of subjective feedback only, lacking any prototype or tool and objective performance measures.

This chapter consists of two sections: the first summarizes a small subjective selection of relevant studies¹ and articles which did not introduce a new tool which was tested. These papers are described in greater detail. The second section summarizes many tool-oriented approaches which influenced the research field within the last decades. Boardman and Sasse (2004) used the terms *empirical studies* and *prototype design*. The studies which propose and evaluate a new prototype, however, also include empirical data which was collected. In this thesis, these two categories are called PIM *strategies* and PIM *tools*.

1. In W. P. Jones and Teevan (2007) there is an exhaustive summary of many projects up to 2007.

2 Research Background

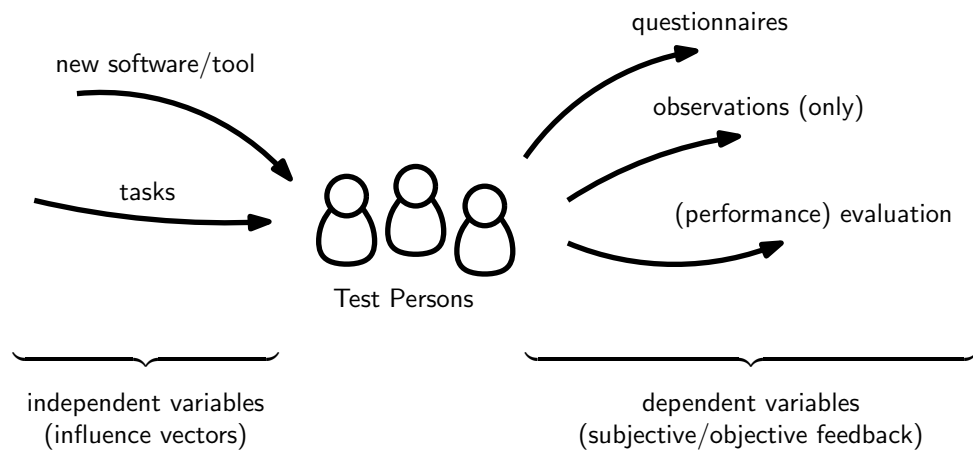


Figure 2.1: Research contributions can have or make use of influence on test persons. For subjective or objective feedback results, questionnaires, passive observations, or (performance) evaluations are being used.

2.1 PIM Strategies

This section summarizes research work focused on the status quo of user strategies without introducing new methods or tools.

2.1.1 As We May Think

The famous article “As We May Think” (Bush, 1945) was a milestone in the history of modern information management. Vannevar Bush summarized the status quo and proposes new goals for the world of science after the end of World War II. He took a look at the history of technology and described specific problems of famous scientists which limited them. At the time, they did not have fabrication standards and appropriate tools for building inventions. He stated that “[m]achines with interchangeable parts can now be constructed with great economy of effort”. Bush proposed a research focus shift from problems related to the physical world towards a new direction.

2.1 PIM Strategies

In his opinion information storage, processing and retrieval represented very important challenges. He then described many state of the art techniques of the time and extrapolated them. His descriptions could be interpreted to understand that he anticipated things like digital cameras, fax machines, the advantages of digital copies, text-to-speech conversion systems, optical character recognition, the availability of general-purpose computers for everybody, modern programming languages, relational databases, non-cash payment methods, magnetic information storage systems, brain-computer interfaces, the digitization of information of any kind, and so forth.

The most revolutionary thing he mentioned, by far, was an apparatus he named “memex” (probably for “memory expander”). It was “a future device for individual use, which is a sort of mechanized private file and library [...] in which an individual stores all his books, records and communications”.

The physical appearance of a memex was described as a desk which “can presumably be operated from a distance”. One could operate this device using a keyboard, buttons and levers. Material was stored on microfilm. The storage capacity was so large that “it would take [the user] hundreds of years to fill the repository, so he can be profligate and enter material freely”. Content could be purchased from third parties and easily inserted into a memex. Bush clearly described the process of copy and paste, skimming and navigating large documents and annotating content.

As the “essential feature of the memex”, he emphasized the possibility of being able to create links: “the process of tying two items together is the important thing”. This way, memex is one of the first concepts describing hyperlinks. Bush also described how joined items form a trail, where items “from widely separated sources [are] bound together to form a new book”.

This vision of Vannevar Bush was quite remarkable, considering the time when it was written. Bush described visions of techniques and tools of an information worker and not of an average clerk or scientist of that time. This single article inspired many scientists in the following decades. Not all of his dreams became products yet. One important idea which is also a very crucial point for this thesis, Bush described that way:

2 Research Background

“ Our ineptitude in getting at the record is largely caused by the artificiality of systems of indexing. When data of any sort are placed in storage, they are filed alphabetically or numerically, and information is found (when it is) by tracing it down from subclass to subclass. *It can be in only one place*, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome. Having found one item, moreover, one has to emerge from the system and re-enter on a new path.

The human mind does not work that way. It operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts, in accordance with some intricate web of trails carried by the cells of the brain. It has other characteristics, of course; trails that are not frequently followed are prone to fade, items are not fully permanent, memory is transitory. Yet the speed of action, the intricacy of trails, the detail of mental pictures, is awe-inspiring beyond all else in nature.

[Emphasis by Karl Voit].

2.1.2 How People Organize Their Desk

In 1981, Malone (1983) interviewed ten persons who gave a “tour of their office”. He asked a set of standard questions to find out how people organize their physical desk and office in general. This exploratory study revealed that people can be classified into those having a “neat office” and those having a “messy office”. People having neat offices tended to work using routine workflows. They were filing documents into a well-defined organization scheme, which often reflected these (externally-defined) paper workflows. Interviewees having a messy office often relied on piles of mixed content. The two most important units of desk organization were files and piles. Malone emphasized that “in addition to this *finding function*, an equally important function of most desk organizations is *reminding*”. He found evidence that there were cognitive difficulties in classifying information.

2.1 PIM Strategies

Most interesting were the implications Malone derived for (future) “computer-based information environments”. Computer systems should support the mechanical process of creating multi-leveled classifications and labels. Users should be able to defer classification, without the need for having an explicit title. Another important feature was the notion of “accessing information on the basis of its spatial location instead of its logical classification”. Automatic classification was a way to simplify classification. Explicit and implicit information should be used to support this automatic process. The temporal view was an important aspect of how information was accessed by users. Retrieval should be supported by more than one dimension at a time. In relation to the reminding function, he stated that “systems should make it easy for their users to store certain information so that it will automatically appear, without being requested”. When a user explicitly defines priorities, the system should make use of the following indications: (a) the frequency of being displayed, (b) the size of its graphical representation, (c) the more or less prominent location of its representation, and (d) the color of its representation. Malone also suggests “that computer-based systems can automatically change priorities over time”. According to its “semiintelligent reminding system”, the computer should be able to change priorities.

2.1.3 The Psychology of PIM

Another great insight into the topic of PIM was given by Lansdale (1988). This paper summarized the psychological aspects of recall, recognition and categorization. Due to its direct relevance to this thesis and the fact that this paper discussed many issues that modern computer interfaces still do not address, the description of its content will be in greater detail.

Lansdale initially stated that “the purpose of IT should be to increase the quality, not merely the quantity, of available information”. He mentioned “two general points [...] which represent important issues in the development of information management tools on computers”:

- “ First is that there is a general problem in categorising items, both in terms of deciding which categorisations to use, and in

2 Research Background

remembering later exactly what label was assigned to that categorisation. The second is that we remember far more about documents than can be used in retrieval procedures. Clearly, if the first of these could be ameliorated, and the second exploited, powerful tools will emerge.

He further refers to the study done by Malone:

“ Let us consider the use of piles identified in Malone’s study. No one would suggest the introduction of unstructured “piles” of documents in a computer environment. (I say this with the thought that somewhere someone probably has, much in the way that someone thought of building planes that flapped their wings.) Apart from anything else, they are evidently counter-productive. *How, then, do we decide which aspects of office behaviour to emulate in office automation and which to avoid?*

The problem is that the *strategies used by people in one technology need not apply to another.*

[...]

The point is subtle but critical to a psychological analysis of information management. *The piles that Malone reports are not, in a simple sense, representative of a need in the user.* Quite the reverse, in fact. They are a compensating strategy for the problems of classification. In using piles, the worker is making a trade-off along several dimensions of difficulty. *[Emphasis by Karl Voit].*

This introduced a remarkable notion to PIM: how can researchers decide which user behavior is expressing a need and which behavior is simply a workaround for an inappropriate tool environment?

“ [T]he underlying psychology of information management cannot be directly inferred from users’ behaviour in offices, because that behaviour is largely adapted to overcoming the problems being created by the mismatch between the facilities provided, the users’ need, and their cognitive capacities.

Lansdale criticized the unreflected emulation of concepts from the real world in their virtual counterparts. He questioned fundamental assumptions of the [desktop metaphor](#):

“ For the moment, I will leave the reader with the thought that concepts such as in-trays and filing cabinets, artefacts of a paper-based technology, may transfer sensibly to a computer-based system, because they have a genuine function and fulfil a cognitive need. On the other hand they may be the trappings of a constrained and outmoded technology which have no relevance in computer-based systems.

In another part of the paper, Lansdale mentioned:

“ It is important to recognise that the design philosophy of these machines [based on the desktop metaphor] is not principally to support information management, but to make the machines easy to use.

[...]

However, in areas of information management which involve longer term storage and retrieval, their facilities do not provide added value. When it comes to filing, these systems resort to rather traditional methods: the files need a name, they can be placed in a folder, and in a particular “filing cabinet” or on a particular disc. Ultimately, to retrieve this information, we are back to remembering filenames and the categorisations we used to file the information. The added visual aspects of the interface do not provide any help here: icons of a particular type (such as documents) all have exactly the same appearance. The only way they are differentiated is by a filename underneath them.

The paper further concentrated on the problems of classification and the role of memory. Since the 1980s, many information interfaces were developed, where the software designer selected a particular category for each menu entry. But “[u]sers were making too many mistakes by way of selecting the wrong category on the menus”. Lansdale further concluded that “information does not fall happily into neat categorisation structures”.

2 Research Background

Furthermore, “information in the real world falls into several overlapping and fuzzy categories, which means that any categorisation of an item of information can only be relevant to certain aspects of it, even if it can be used accurately”.

One of the most important conclusions from Lansdale (1988) was that

“ The process of information retrieval in the human mind is fundamentally different from filing or library systems in which items are accessed by location rather than by their meaning.

This questions the desktop metaphor in its fundamental roots. Even though this metaphor is helpful during the learning process of a computer interface, its role in making this computer interface an efficient tool is disputable. Lansdale emphasized this notion with a general statement regarding transferring results from one technology to another:

“ Consequently, if the technology were to change, and with it the constraints, then the match between the psychological processes and the technology would result in different trade-offs. Consequently, strategies would change also, and *what constituted a reasonable strategy in one technology need not in another.*

A similar argument applies to the interpretation of psychological experiments. Just because particular experiments on the memory for pictures appears to produce high levels of recall, it does not follow that tasks involving pictorial memory will always do so. It could be that the particular circumstances of the experiment allowed for successful strategies which may not be applicable elsewhere. *[Emphasis by Karl Voit].*

Many GUIs use visio-spacial methods in order to make use of the location-based memory capabilities of computer users. Lansdale came to the conclusion that “the subjects do not perform better with visio-spatial information”:

“ [A]ssertions made about the inherent value of visio-spatial information represent a simplistic view of human cognition and no guarantee of good design. [...] [I]t is important to emphasise

2.1 PIM Strategies

that this does not mean that visio-spatial information should be dismissed in considering future information systems. [...] What is being dismissed is that visio-spatial methods are automatically a panacea for human-computer interaction.

Regarding the large set of studies that showed a remarkable recall using visio-spatial methods, Lansdale explained that “the success of recall is more probably related to the particular circumstances under which they are being asked to perform than resorting to concepts of supernormal psychological powers”.

The paper tried to give some answers regarding the question of how filing should look like. However, Lansdale found out that there are trade-offs to be made:

“ This leaves us with an important dilemma. The more we ask the user to do at the process of information storage, the less likely he is to do it, creating retrieval problems. On the other hand, the more we automate the process of storage and take responsibility away from the user, the less he is going to remember, and therefore the less he is going to be able to retrieve.

2.1.4 Cross-Tool Behavior and Longitudinal Data

In Boardman and Sasse (2004) the authors “collected cross-tool data relating to [file](#), email and web bookmark usage for each participant, and [...] [they] collected longitudinal data for a subset of the participants”. Owing to its high production effort, longitudinal data is seldom part of PIM study results. However, this longitudinal data shows very important insights into how people respond to different situations over time. Additionally, PIM studies usually concentrate on a single method or tool ignoring differences and side-effects resulting from other methods or tools.

The study was conducted in two separate phases. In the first phase, “semi-structured interviews with 31 users centered on guided tours of their file, email and bookmark collections on their main work computer” were carried out:

2 Research Background

“ We performed content analysis on the interview data to extract themes relating to strategies, problems and needs. Screenshots were also captured of the desktop, and the folders in each collection. We analyzed the folder structures to investigate: (1) the concepts used to name folders (e.g. *project*, *contact*, *place*), and (2) the level of *folder overlap* (folders common to multiple collections). Finally, we compared each participant’s strategies between the three collections to investigate the consistency of their behaviour [...].

In phase two, eight participants from phase one were tracked to get insight on “the evolution of the three collections and the strategies used to manage them”. The average participation duration was 286 days. Additionally the participants were asked to try a new tool called *WorkspaceMirror*. Due to the relatively low impact of *WorkspaceMirror*, its implications will not be mentioned here. *WorkspaceMirror* itself will be described in Section 2.2.8.

One finding of phase one – that is confirmed in many other studies as well – was that the following:

“ Participants were highly motivated to talk about PIM – it was an area that was important to them, and a source of problems and frustration.

So it seems that there is a readiness to talk about problems of PIM among users. My own anecdotal evidence also reconfirms this inherent desire of users to talk about PIM issues.

Related to the topic of this thesis, another result is quite interesting. People seem to have a much greater personal relation to their file collection than to their emails or bookmarks. This is also reflected by the likelihood of information re-use:

“ Our qualitative data suggests that users are more likely to re-use files than emails or bookmarks, particularly over the long-term. Users perceive that file organization is more worthwhile

2.1 PIM Strategies

since the cost of filing is offset by predicted benefits at retrieval time.

Boardman and Sasse tried to categorize participant behavior. The classic patterns of *filers* and *pilers* (Malone (1983), physical desktop) or others² were insufficient to explain participants behavior. The majority of people employed a mix of different strategies. Boardman and Sasse wrote that these classic strategy classifications “exaggerate the extremes” and do not reflect the more complex mix of strategies users develop: “Our cross-tool data indicates that PIM strategies also vary significantly between tools for many individuals.”

Barreau and Nardi (1995) did not try to classify user behavior. Instead, they derived a classification scheme with regard to the information which is processed: *ephemeral*, *working*, and *archived*.

However, Boardman and Sasse proposed two alternative sets of terms:

1. “Information usefulness: *active* (including ephemeral and working), *dormant* (inactive, potentially useful), *not useful*, and *un-assessed* (e.g. new emails).”
2. “Information ownership: *mine* (including self-created files, and items that have been assessed as having value, e.g. filed email), and *not-mine* (e.g. much of the email inbox, and information on the internet).”

The overlap in terms of similarly-named *folders* between *file systems* and email was significant. The overlap between files and bookmarks or emails and bookmarks was much lower.

Boardman and Sasse also reported “a strong preference for browsing over search in all three tools”. This was confirmed in many other studies as well and will be discussed in Section 3.4.2 in more detail.

The authors also developed a system to profile cross-tool behavior of their test users. The importance of files is also reflected in these results as shown in Table 2.1.

2. *Frequent filer*, *spring cleaner* and *nofiler* (Whittaker and Sidner (1996), Emails); *no-filers* consists of *folderless cleaner* or *folderless springcleaner* (Bälter (1997), Emails); *no-filer*, *creation-time filer*, *end-of-session filer*, and *sporadic filer* (Abrams, Baecker, and Chignell (1998), bookmarks)

2 Research Background

Cross-tool profile	# Users	% Users
pro-organizing in all 3 tools	8	26 %
pro-organizing in files & email only	14	45 %
pro-organizing in files only	7	23 %
organizing-neutral in all tools	2	6 %

Table 2.1: Cross-tool profiles. [Source: Boardman and Sasse (2004)]

The study had effects on some of the participants' strategies: two persons (out of the eight) changed their PIM strategies during phase two of the study. The authors mentioned a "perceived social pressure" to be organized which might be the reason to change strategies.

“ Although the observed changes were subtle, participants found them beneficial. However, the supporting nature of PIM means that users rarely devote time to planning and executing changes in strategy. Users may benefit from increased reflection with respect to PIM, so as to receive the same benefits that resulted from the “self-auditing” effect of the study.

Boardman and Sasse (2004) contained a quite remarkable statement related to the usage of [folder hierarchies](#):

“ The folder hierarchy is often criticized for not being easily adaptable to fast-changing user needs, and requirements for dynamic views of personal information are often emphasized in PIM design [...]. Our findings suggest a contrasting perspective: the slow-changing nature of the hierarchy may benefit users by promoting familiarity with the personal information environment. Such familiarity in turn supports location-based finding for which users expressed a clear preference. We thus highlight persistence as an often overlooked, yet desirable design goal.

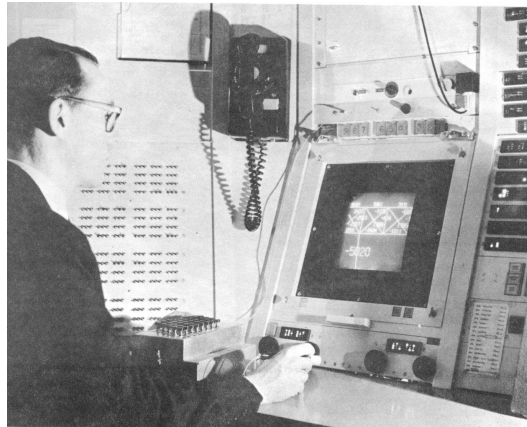


Figure 2.2: “TX-2 Operating Area – Sketchpad in use. On the display can be seen part of a bridge [...]. The Author is holding the Light pen. The push buttons used to control specific drawing functions are on the box in front of the Author. Part of the bank of toggle switches can be seen behind the Author. The size and position of the part of the total picture seen on the display is obtained through the four black knobs just above the table.” [Source: Sutherland (1963b)]

2.2 PIM Tools

In contrast to the last section, the research work listed in this section relates to new methods and research prototypes which were evaluated. Starting with two very early milestones, [Personal Information Management \(PIM\)](#) tools developed over time with very different approaches.

2.2.1 Sketchpad

In 1963, Ivan Sutherland (Sutherland, [1963a](#); Sutherland, [1963b](#)) designed and implemented ground-breaking drawing software on a Lincoln TX-2 computer at MIT. Its interface (Figure [2.2](#)) was the first direct manipulation computer interface and resembled an ancestor of modern CAD tools. Instead of using punchcards or typed text as user input, Sketchpad used a light pen which was held by the operator like an ordinary pen.

2 Research Background



Figure 2.3: Douglas Engelbart during his demonstration of NLS. The variable split screen view allowed the audience to see the performer and the screen content. In this sequence, Douglas Engelbart showed a graphical representation of a travel plan which was enriched with hyperlinks on each node. For example, a hierarchical shopping list, or books at the library. *[Video Screenshot]*

Besides its direct influence on GUI-design, Sketchpad introduced the concepts of object-oriented programming, templates, drawing libraries, copy and paste, constraint satisfaction and many more.

2.2.2 NLS

A mind-blowing demonstration was conducted by Douglas Engelbart in 1969, presenting his NLS described in Engelbart and English (1968). This demonstration used an enhanced video projection system, showing images from various cameras in San Francisco and Menlo Park and CRT screen content from multiple NLS terminals as shown in Figure 2.3. Multiple researchers were connected with audio headsets. This video-conference system was used to present NLS running remotely on computers connected through telephone lines.

The NLS system itself introduced the computer mouse for interactive pointing and selecting objects on the screen. Hypertext links within text files and

graphical drawings could be used to generate a heavily interlinked documentation system. It included a word processing system, which was used to generate the paper of Engelbart and English (1968). Data could be retrieved by navigating the link structure or by using dynamic search queries with weighted result sets. In its core, NLS was a collaborative multi-user system. Several people were able to collaborate within files and send messages to each other. All actions were logged and enriched with metadata.

This presentation made such a huge impression that people now refer to it as “the mother of all demos”.³

2.2.3 Lifestreams

In the mid-1990s, Eric Freeman and David Gelernter described Lifestreams (Freeman and Fertig, 1995; Freeman and Gelernter, 1996; Freeman, 1997), a client-server software which organized “a user’s personal workspace” using “a time-oriented stream of documents”. Its premises – as described in Freeman and Gelernter (1996) – were:

1. Storage should be transparent. “Naming” a file and choosing a location for it as it is created is unnecessary overhead.
2. Directories are inadequate as an organizing device.
3. Archiving should be automatic.
4. The system should provide sophisticated logic for summarizing or compressing (and where appropriate, for picturing or animating) a large group of related documents of which the user wants a concise overview.
5. Computers should make “reminding” convenient.
6. Personal data should be accessible anywhere and compatibility should be automatic.

Lifestreams visualized all documents a user created and all documents which were sent to the user in a stream, as shown in Figure 2.4. The user was able to create new documents by using the *new* and *clone* functions. The latter “takes an existing document, creates a duplicate and adds it to [the]

3. <https://www.youtube.com/watch?v=yJDv-zdhzMY> – retrieved on 2013-02-16.

2 Research Background

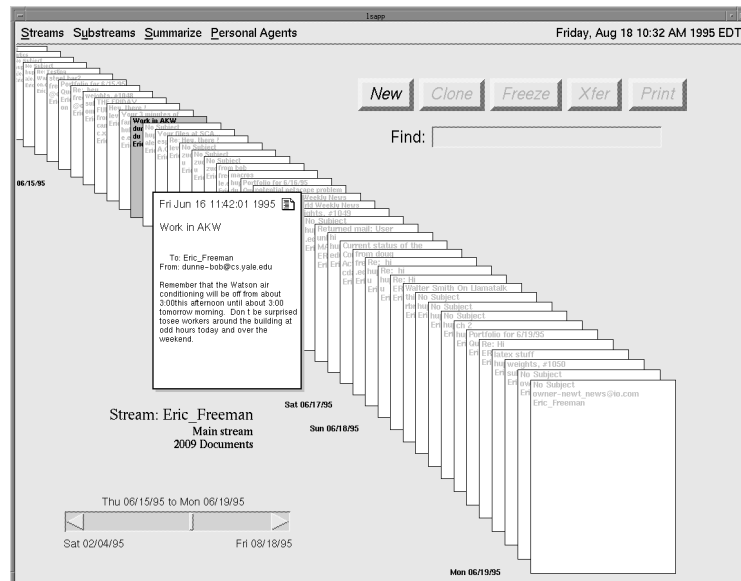


Figure 2.4: Lifestreams interface, UNIX Viewport. [Source: Freeman and Fertig (1995)]

stream”. Further on, “[d]ocuments can also be created indirectly through *transfers*, which copy a document between streams”.

For re-finding documents, Lifestreams offered a *find* operation. The result sets of such queries were shown as sub-streams and behaved like virtual folders: changes to the result of the query were visible instantly.

The *summarize* operation generated an overview document from a given sub-stream. This supported visualizing stock prices and so forth. To change a document from a writable state to a read-only state, the *freeze* operation could be used. Using an *xfer* operation, a user was able to send documents to other users using email.

A template module provided an easy-to-use method for making phone call notes, storing business card information, or logging times while working on a task. External data sources were added using web bookmark items, which could also be shared and transferred among multiple users.

Lifestreams was evaluated in Freeman (1997) using the [Questionnaire for User Interaction Satisfaction \(QUIS\)](#), prototype instrumentation, and a user response survey. Six users participated in the evaluation. Freeman found that subjective user acceptance was very high and that Lifestreams had a short learning curve. The features of Lifestreams offered a clear benefit for the test users. Unfortunately, Lifestreams is not available for download or evaluation.

2.2.4 TimeScape

A similar approach for visualizing documents in a time-oriented view was described in Rekimoto (1999). A research prototype called TimeScape resembled a time machine (similar to the “Time Machine” feature of the current os x). Users were able to put icons on the desktop and arrange them arbitrarily. All user activities were logged to restore desktop states from the past. This way, users were able to delete documents from the desktop and reach back to them using the time machine. The duration of an object was defined from its first appearance on the desktop until its delete time-stamp from the desktop. When a user went to the future, she was able to put sticky notes on the desktop which acted as a reminder. When the future time was reached, this sticky note appeared on the desktop. TimeScape offered zoom-in and zoom-out on a time-line scale, which successively faded away objects that had shorter durations.

The concept was completed with a modification of the Samba [file system](#) called “TmSamba” and a protocol to communicate between applications called “time-casting”. The authors proposed that every application should be able to change its state according to the time selected by TimeScape.

To my knowledge, TimeScape and its companion tools were never released and never evaluated outside the development team.

2 Research Background

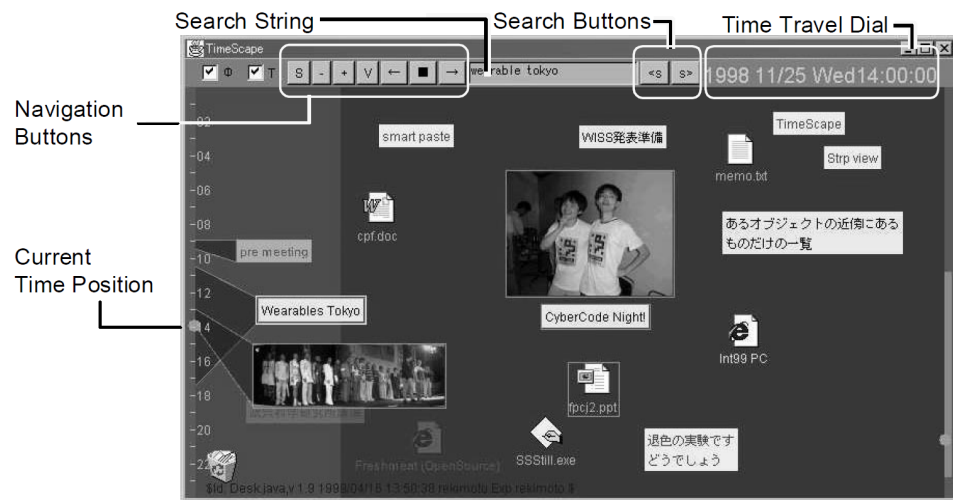


Figure 2.5: A screenshot of the TimeScape desktop.

2.2.5 Remembrance Agent

The Remembrance Agent (as described in Rhodes and Starner (1996); Rhodes and Maes (2000); Rhodes (2000); Rhodes (2003)) is a mode in the Emacs editor. The editor interface becomes split, so that the current working buffer is shown on top, whereas the narrow Remembrance Agent buffer is shown on the bottom of the window (Figure 2.6). By default, the Remembrance Agent buffer contains four lines filled with suggested links to relevant documents such as emails, files or paper abstracts. It is updated every few seconds and relates to the content of the main working buffer above.

The user has several interaction possibilities: get the list of keywords that lead to the suggested link, retrieve the full text of the suggested information, use Remembrance Agent as a normal search engine, switch the databases to be queried, and associate specific sets of databases to a certain file type.

The evaluation described in Rhodes and Maes (2000) was a controlled task evaluation. Twenty-seven persons were divided into a control group and an experimental group. All users were asked to write a newspaper-style article about a subject. The control group had to use a normal Emacs editor and

2.2 PIM Tools

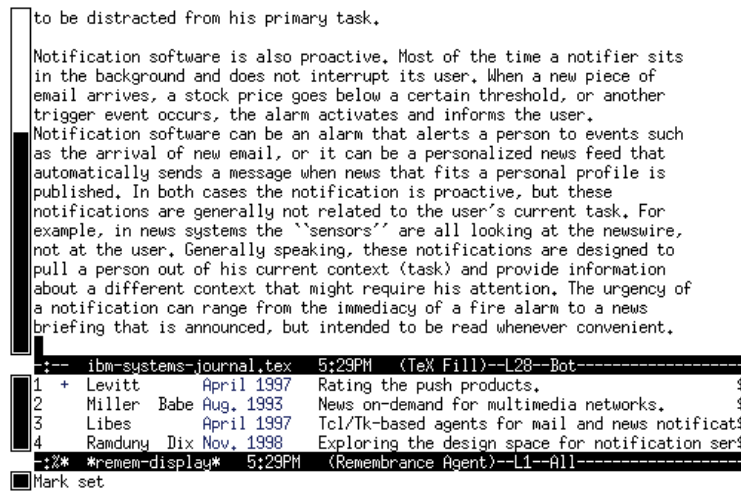


Figure 2.6: A screenshot of Emacs showing the Remembrance Agent giving context-related links to relevant information. In the upper section, Emacs is showing the current working buffer. In the four lines of the lower section, the Remembrance Agent shows links to related resources. [Source: Rhodes and Maes (2000)]

a web browser⁴ to search within an archive of news articles. The users in the experimental group were given a version of Emacs enhanced with the Remembrance Agent, whose index was fed by the newspaper database. In addition, the experimental group was able to use the web search as well.

Results showed that a large majority of the users preferred the Remembrance Agent over web search “in terms of ranking, overall usefulness, and whether they would want the tool for a similar task”. And “subjects from the experimental group viewed around three times as many different Tech articles as did those in the control group, and within the experimental group subjects viewed around two-and-a-half times as many articles using the [Remembrance Agent] as they did using the search engine”.

The produced essays were examined as well: “Articles were blinded and

4. To be precise, the web browser was enhanced with another research tool called *Margin Notes* which automatically annotates web pages. Besides the fact that *Margin Notes* performed poorly, its influence was minimal and is left out in the further description here.

2 Research Background

coded for number of facts mentioned, number of references to [the article database], and overall quality. However, individual variance was high, and no statistically significant difference was found between the two groups.”

In short, users found Remembrance Agent to be a useful tool, which supported their task by giving them links they regarded useful.

Remembrance Agent was released under the GPL and its source code is released on a web page⁵. Version number 2.12 was published on February 16 2004 for Emacs-20 and XEmacs-20 and is meanwhile considered unsupported and outdated.

2.2.6 Presto

The research software Presto was developed (Dourish et al., 1999a; Dourish et al., 1999b; Dourish, Edwards, Lamarca, et al., 2000; Dourish, 2003) within the Placeless Documents project. Presto targeted the document as the most important entity, where properties⁶ played a central role in the design. Those properties were associated directly to the documents and moving a file did not break these relations. Files were organized without any hierarchical structure. Instead, collections could be used to group documents.

Collections held documents which met collection-properties the user specified. The latter ones were comparable to Smart Folders used in Mac os x Finder. Additionally, the user was able to include arbitrary documents in a collection that did not meet the properties for the search query (inclusion list). Similarly, the user was able to manually remove documents from a collection (exclusion list).

Many implicit properties were collected using so-called “services”. For example, an email service extracted sender, subject, date and so on from email headers. Legacy applications could access documents using a mounted NFS share.

5. <http://www.remem.org/> – retrieved on 2012-07-14.

6. Properties in Presto were key/value pairs such as “author=dourish” or “status=draft”.

2.2 PIM Tools

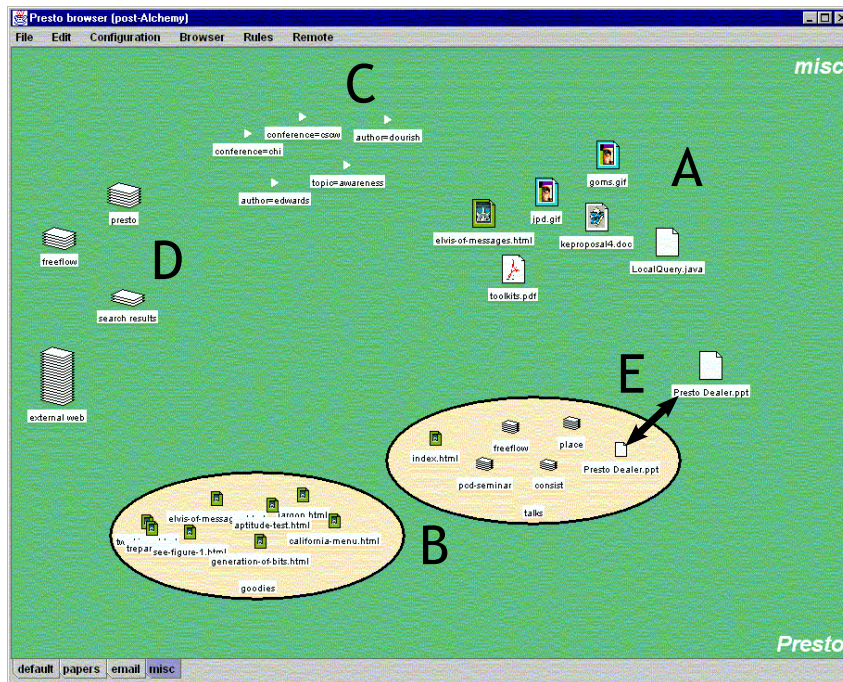


Figure 2.7: “A snapshot of Vista, a Presto interface. Vista offers multiple Rooms-like workspaces onto the same document database, and displays documents (A), collections (B) and properties (C). Closed collections are shown as piles (D), giving cues as to their current size. Note that the nature of the system implies that a document may actually appear in multiple places at once (E).” [Source: Dourish et al. (1999b)]

For visual access to Presto, a workspace browser named Vista was developed. Its GUI (Figure 2.7) used mouse operations to interact with the user.

To my knowledge, Presto was never evaluated in studies or released to the public.

2.2.7 Haystack

The Haystack project (Huynh, Karger, and Quan, 2002; Huynh, Karger, Quan, and Sinha, 2003; Bakshi and Karger, 2005; Sinha and Karger, 2005)

2 Research Background

at MIT implemented a concept that was to some extent similar to the Presto project. Unlike Presto, Haystack did not see a file as the sole information object. Using technologies like [Resource Description Framework \(RDF\)](#) from the [semantic web](#), it provided methods to create links between objects and annotate them. Objects were derived from files automatically or could be added manually.

A series of components was developed in the Haystack project. As described in Sinha and Karger (2005), a task workspace was presented, allowing an information worker to set up her environment:

“ We define a task workspace as a collection of information relevant to the task at hand that can be selected, presented and operated upon based on the user and task constraints, and that the user can aggregate in a lightweight manner. Our approach combines three elements:

- A *workspace designer* that lets users lay out the sets of information objects they want to work with in their application, specify which view to use to present each type of object and stipulate the relevant operations that should be readily available;
- A *view designer* that lets users specify how each type of information object in their workspace should be shown – what properties of those objects they want to see, how users should interact with them, and how they should be laid out; and
- A *channel manager* that lets users specify queries that dynamically maintain collections of related information that can be used to specify the relevant sets of information for one or more task workspaces.

With those components, users were “able to define and modify the content, presentation and manipulation capabilities of their task workspaces” as shown in Figure 2.8. These views were data themselves and could be named, re-visited, re-used and linked. This way, a user was able to share views and quickly switch from one workspace to another according to the current task.

2.2 PIM Tools

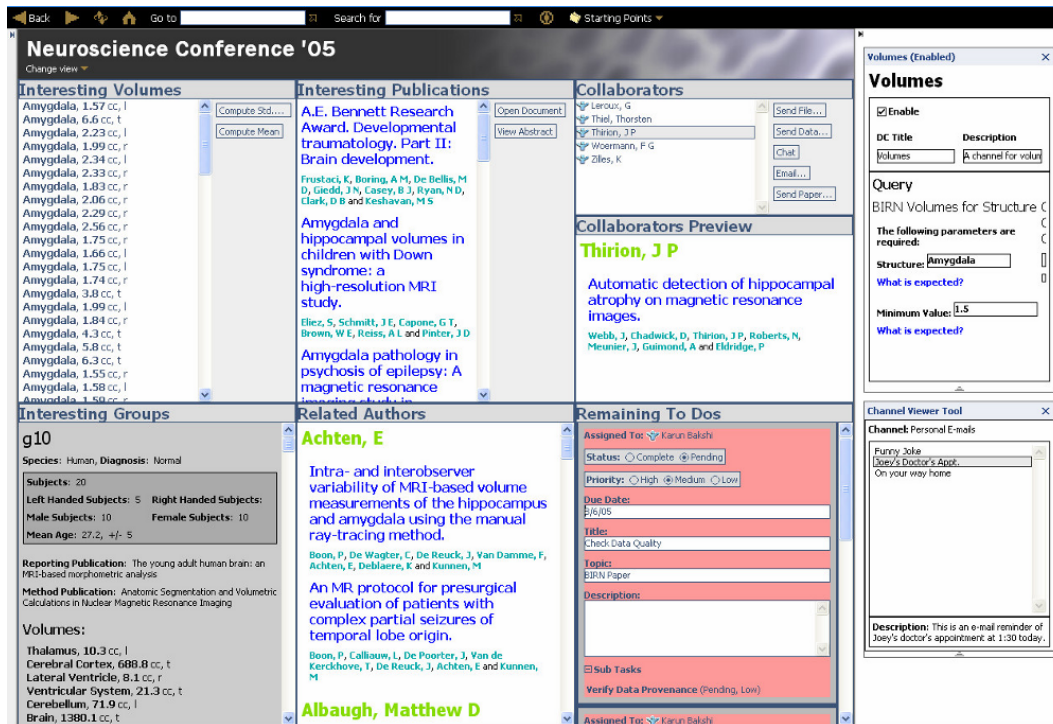


Figure 2.8: Haystack: paper-writing workspace in usage mode. The user was able to define the layout of the information object sources, the type of information objects shown, and to define the queries for information objects shown. [Source: Bakshi and Karger (2005)]

In contrast to wide-spread relational database user interfaces, “Haystack aims to feel less like a database, and more like an application” as W. P. Jones and Teevan (2007, p. 149) stated.

The source code of the Haystack project is released⁷ under a BSD license. At the time of writing, however, the Subversion (svn) repository was broken.⁸

7. <http://simile.mit.edu/hayloft/> – retrieved on 2013-02-16.

8. Error message “No such revision 6206” on the web interface and when trying to check out the svn source as well.

2 Research Background

2.2.8 WorkspaceMirror

Boardman and Sasse (2001) and Boardman (2001) described an overlap in folder-names when analyzing [file system](#) folders, email folders, and web bookmark hierarchies. The most noticeable amount of overlap was found between file system folders and email folders.

Boardman supposed that a tool which automatically synchronizes the organization of folders between file system, email, and bookmark could improve PIM: users would have a supporting tool for consistency and worry less about re-creating similar structures in different tools. *WorkspaceMirror* (Boardman, Sasse, and Spence, 2002) was developed which was described as such:

“ Our current prototype is an extension to MS Windows and synchronizes three folder hierarchies: (1) email folders stored in MS Outlook, (2) “My Documents”, used to store personal files, and (3) web bookmarks folders under “Favorites”. The tool works in one of two modes: automatic or prompted. In prompted mode the creation, deletion or renaming of any folder causes a dialog box to be displayed asking the user if they want to replicate the operation in the other two tools.

A preliminary evaluation with four of their colleagues showed promising effects with three of them welcoming *WorkspaceMirror* and one preferring different hierarchies.

In Boardman and Sasse (2004) (which was covered in detail in Section 2.1.4) a long-term study revealed that only four participants used *WorkspaceMirror* over a longer period of time (average: 107 days). The other four participants could not find enough advantages to use it long-term. In general, the participants did not see *WorkspaceMirror* to be of notable help:

“ Three “non-changers” used *WorkspaceMirror*, mirroring 14 new folders on average, mostly between files and email. We had anticipated that *WorkspaceMirror* would stimulate pro-organizing strategy changes by allowing users to leverage filing investment in one collection across to other collections. However, these

participants instead employed WorkspaceMirror in support of their existing filing strategies. Against our expectations, the two participants who made the most significant strategy changes did not consider WorkspaceMirror to be a major contributory factor.

Although WorkspaceMirror was a very interesting cross-tool approach, it seems to be the case that the overlap in structure is only a minor one. Users seemed to prefer different kinds of structures in different tools. Boardman and Sasse (2004) summarized:

“ Our observation of folder overlap points to a subset of user activities that involve the management of multiple types of information. Most overlapping folders corresponded to *roles* and *projects*, suggesting that these concepts may be usefully shared between collections [...]. However, it should be emphasized that most folders did *not* overlap. This suggests that: (1) some production tasks are supported by single PIM tools and may not necessarily benefit from increased integration; and (2) users may have different organizational needs in different tools. In addition our data indicates that email contains more *contact*-based folders, whilst bookmark folders are mainly *interest*-based. This variety suggests users may be constrained by integration designs that are based on specific types of concept [...].

To my knowledge, WorkspaceMirror was never released to the public.

2.2.9 Stuff I've Seen

Another excellent research contribution which combined a proposal of a tool with a long-term study was Dumais et al. (2003). The authors developed a new client desktop search engine interface called *Stuff I've Seen* (*Stuff I've Seen* (sis)). Built on top of Microsoft Search indexing⁹, sis sorted results by last time modified or an Okapi-based ranking algorithm. It allowed facet-based refinement through filters and showed document title, date, rank, author, and so forth. For emails, the recipient was determined as

9. http://en.wikipedia.org/wiki/Windows_search – retrieved on 2012-07-28.

2 Research Background

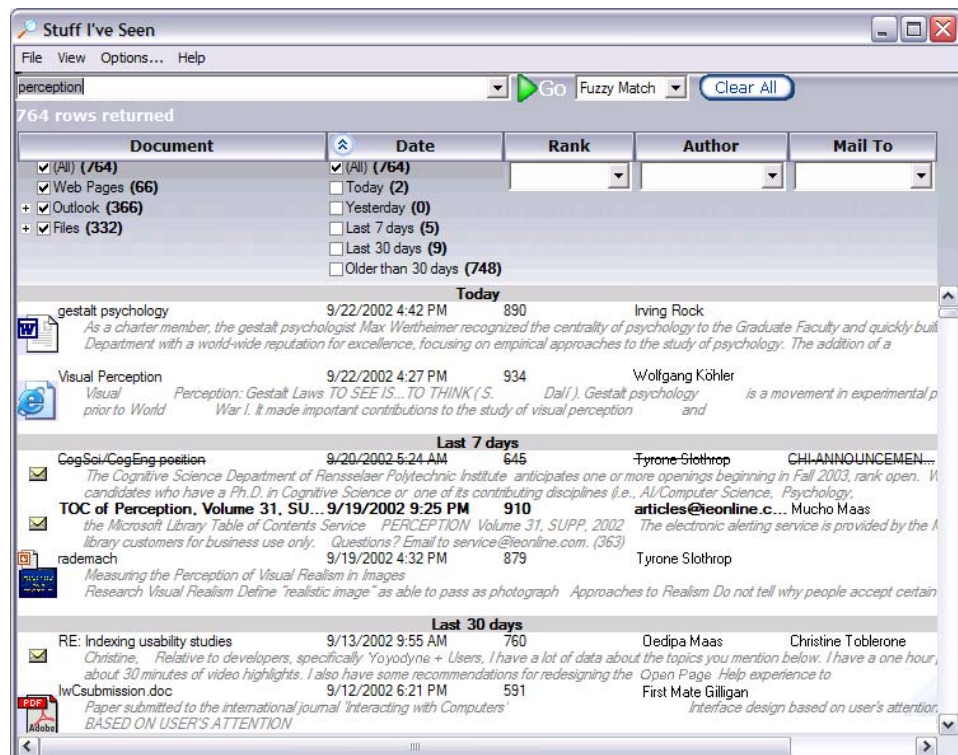


Figure 2.9: The sis interface, with the Top View. On top is the query input box. Underneath are the filters. At the bottom the matching results are shown. [Source: Dumais et al. (2003)]

well. Additional fields (File Type, Mail CC, Mail HasAttachment, Message Type, Message Read, **Path**, Size) were available through an option menu. For images and PowerPoint slides, a thumbnail preview was shown. For documents containing text, the first 300 characters were visible. This way, the authors achieved a set of rich contextual cues.

The GUI of **sis** had two different visualization layouts: the Top View as shown in Figure 2.9 and the Side View as shown in Figure 2.10. The Top View was considered more flexible in terms of expressing a wider range of filter criteria. The Side View “[had] the advantage that it [was] somewhat easier to understand and [was] less cluttered”.

Dumais et al. evaluated [sis](#) with 234 participants over a period of six weeks. The participants had to fill out a pre-test questionnaire and a longer questionnaire after one month of usage. Usage logs were captured by the [sis](#) version used.

The interface was delivered in four different default configurations:

1. Top View, ordered by date.
2. Top View, ordered by rank.
3. Side View, ordered by date.
4. Side View, ordered by rank.

Participants were able to change the default sort order by themselves.

Log files showed that participants made 4.4 queries a day, but the number had a high variance among users. Like many other studies on search interfaces, the participants did not make much use of Boolean expressions and they used short queries (average of 1.59 words). Due to the good interface design, participants made use of frequent direct filter manipulation.

The queries showed that people were important factors for searching information:



The most common query types in our logs were People/places/things, Computers/internet and Health/science. In the People/places thing category, names were especially prevalent. Their importance is highlighted by the fact that 25% of the queries involved people's names suggesting that people are a powerful memory cue for personal content. In contrast, general informational queries are less prevalent.

It is important, however, to notice that participants used the [sis](#) interface mainly to search for Outlook emails. This was proved by the type statistics of opened items: "Email was by far the most common type opened (76%), followed by web pages (14%) and files (10%)". Queries related to people are possibly more prevalent when people are searching within emails.

As also shown by Gibson, Miller, and Long ([1998](#)) and Leung et al. ([2008](#)), most opened items were not very old. Hence, providing user support for finding the most recent items is an important issue.

2 Research Background

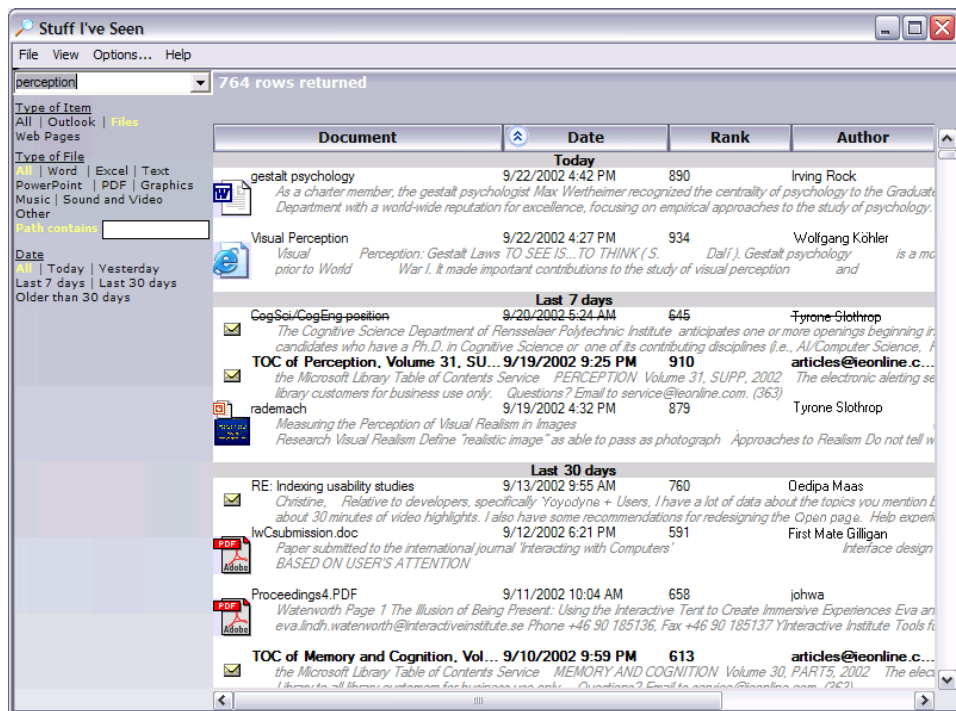


Figure 2.10: The sis interface, with the Side View. The query input box and a simplified version of the filters are located on the left side. The right-hand side is filled with the result set. Compared to the Top View this leaves more space for results and offers simpler and easier use of filters. [Source: Dumais et al. (2003)]

“ Overall, 6.6% of the items opened were first seen that day, 21.9% within the last week, 45.9% within the last month, and 89.4% during the last year. Not surprisingly, recent items are accessed frequently, but the distribution has a long tail with items up to eight years old being opened.

A result set sorted by date was more popular among the participants than a result set sorted by the rank algorithm which was included. This might be a direct consequence of the fact that users mostly tended to access recently modified items.

The questionnaires reported positive reactions from the participants (“Users were overwhelmingly positive about [sis](#)”). Obviously, there was a need for this kind of desktop search interface. From the comments of the respondents the researchers could deduce that “they had trouble understanding different kinds of searches such as fuzzy match and Boolean queries”.

Participants reported the problem of items being wrongly categorized in the storing process:

“ People also said that [sis](#) was helpful for finding things that were “buried” or filed in the wrong location. Even people who regularly file email and documents, often misfile information, or create new folders with slightly different names.

To my knowledge, [sis](#) was never released to the public.

2.2.10 Phlat

As a direct successor of the [sis](#) project described in the previous section, Phlat resembled an enhanced version of [sis](#) with added functionality. Phlat was described in Cutrell et al. (2006). The authors described their goal as “[r]ather than viewing search and browse as separate behaviors, Phlat treats them as two ends of a smooth continuum”.

[Tagging](#) was added to the interface. The same interface was used to apply tags to items and for filtering by applied tags. Tags were stored either as NTFS properties (files only) or as MAPI properties (Outlook/Exchange emails only). Users had to tag everything by themselves and tags were not of any use outside the Phlat interface. There was no tag recommendation system and the paper did not mention any [tag completion](#) mechanism.

All currently applied filters were made visible in the query area in order to make perfectly clear what and which filters are active.

An interesting design decision was reported for the question whether to show the document title or the document file name in the result set:

2 Research Background

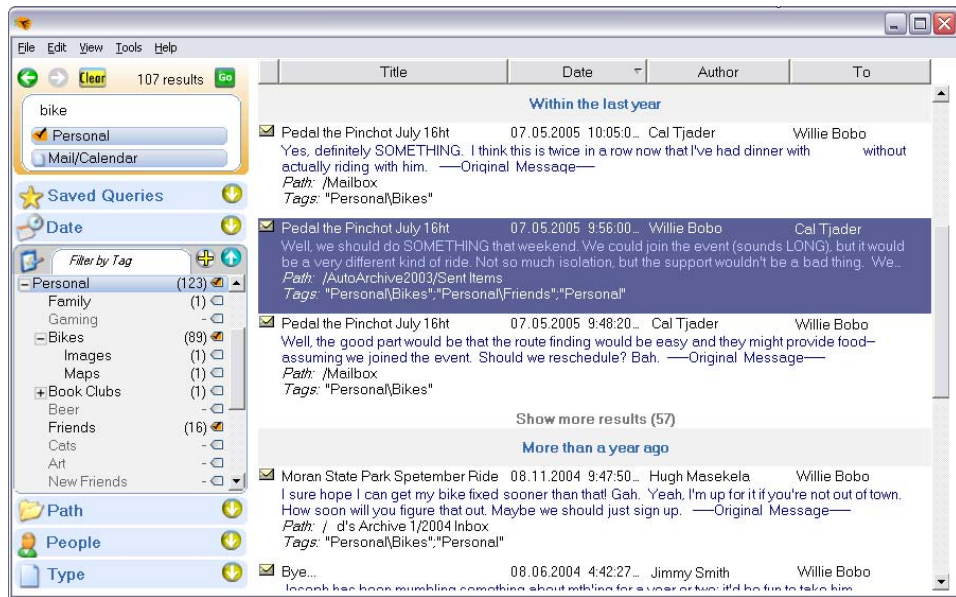


Figure 2.11: The Phlat interface. The interface is quite similar to the Side View of the sis interface. Tagging functionality as well as an enhanced filter overview was added. [Source: Cutrell et al. (2006)]

“ [W]e found that users prefer to see the filename rather than the document title. This is because document titles are often incorrect, misleading or simply empty for many files. Since users typically name their own files, it is the filename that they recognize rather than the document title.

The authors evaluated the Phlat interface with 225 participants over a period of eight months. The study was similar to Dumais et al. (2003) but without questionnaires and without variation of any default settings. Result sets were ordered by date by default.

As an interesting side note, the number of items indexed on participants computers varied significantly:

“ The size of the personal indices of this sample of users varied by almost 2 orders of magnitude, ranging from 5733 items to

more than 472000 items. The median index size was 36182 items.

For log analysis, Cutrell et al. divided log entries into sessions. Those sessions reflected participant behavior to launch several related queries in a short period of time.

“ [W]e defined a session as all queries with an inter-query interval of less than 5 minutes. When the inter-query interval was longer than 5 minutes we checked the semantic content of the subsequent query by comparing the query terms. If it was different, that query started a new session.

Using those kind of metrics, participants use 3.61 queries in an average session. Quite similarly to Dumais et al. (2003), queries were short (1.60 words on average), usage of filters was high, usage of Boolean expressions was very low, and the most popular filters were for persons and item type. Phlat was mainly used for searching in Outlook emails.

The result sets were rather large with an average of 478 entries and a huge standard deviation of 3184. Despite this high number of items in the result set, participants chose to browse further down in the result set, if necessary. In contrast to web search, only 30 percent of all items invoked were within the top three positions.

Within one session, 2.5 items were invoked on average. However, in one quarter of all sessions, participants did not invoke any item at all. This did not mean that those sessions did not bring the desired information. Participants could have spotted the information within the item preview in the Phlat result set. Outlook emails and Outlook appointments were the most common items invoked (80 percent) followed by Word files (six percent). As already mentioned above, Phlat was mainly used for searching within Outlook data.

The use of tagging was quite high: one third of the participants used tags at least at some point during the study. Fifteen percent of all usages of filters involved filtering by tags. This was a remarkable result because other

2 Research Background

studies concluded that users do not want to take on the additional effort of tagging¹⁰.

In contrast to Dumais et al. (2003), Cutrell et al. did not use questionnaires, but the possibility to send feedback via email. This feedback was generally “classified as bug reports, functionality requests, complaints, and complements”. Some of the feedback emphasized the need for new PIM methods:

“ Our users are particularly enthusiastic about the idea of cross-source searching and tagging. We repeatedly heard users delight that they didn’t need to worry about where an item was stored, and many loved the idea of creating a single organizational structure (tags) that spanned email and files (though, as noted above, only a small number of users regularly employed tags).

Some participants reported additional needs for preview views, thumbnails, and better integration into Outlook and Windows. They also wanted better integration into other applications as well. For some file types, such as Microsoft PowerPoint and Microsoft OneNote, the file-based nature of Phlat was too broad. Those file types often contain a larger number of smaller independent information units. But Phlat is only able to tag and filter files and not smaller information units, as Haystack was able to (see Section 2.2.7).

The delay between file updates and index updates was an issue to many participants. File transfers to a non-NTFS file system (like thumb drives) presented another problem, resulting in losing tags. The same thing happened when an email was stored outside of Outlook or forwarded to another person. The authors noted that “if the metadata of tags is too fragile, users will stop spending the time and energy to use them!”

Tagging was mentioned very positively in the feedback emails:

10. Sauermann and Heim (2008): “In general, the approach of the users is to only model when it is necessary and needed later”; Whittaker, Matthews, et al. (2011); ...

“ Despite its many limitations, our tagging system is used quite extensively by a subset of users. Clearly, our users find the idea of universal tagging compelling.

The experience and feedback related to tagging lead to the following statement in the outlook of the paper:

“ Filing will become less important and will be replaced by more general tagging systems.

The authors released Phlat for public download.¹¹ Currently, the most recent downloadable version (1.0.2299.14990, 2006-04-19) is regarded as outdated and discontinued. It is quite questionable whether this version of Phlat could be installed on a current version of Microsoft Windows due to its dependency on old versions of Microsoft Search (up to Windows XP) and lack of maintenance.

2.2.11 Attribute Browser

Marsden and Cairns (2003) tried to surpass the problems of *strict hierarchies* of folders by using a relational database approach. Their Attribute Browser separated the storage format from the access visualization:

“ [W]e believe that the operating system file browser be changed to interact with the file store as dynamically as one can interact with data stored in a relational database. It is not important to the user how the files are stored on disk (performance constraints aside) but the user should be free to choose how those files are viewed and retrieved.

[...]

[O]ur system must present files to a user in a way that lets them perform the following operations in the simplest way possible: restrict, project, union, intersection, difference, group-by and order-by.

11. <https://research.microsoft.com/en-us/downloads/0cdb50f3-ccf6-4198-b874-4643791d4dc4/>
– retrieved on 2012-09-01.

2 Research Background

By using only implicit [metadata](#) such as size, creation date, access date, item type and so on, users were able to filter their items accordingly. In usual file browsers this was only possible within one folder view. Users were able to store queries and combine queries to filter items.

To my knowledge, the research prototype implementation presented in Marsden and Cairns (2003) was never evaluated or released to the public.

2.2.12 MyLifeBits

Following the memex idea presented in Section 2.1.1, Gemmell, Bell, and Lueder (2006) described an approach which originally did not cover PIM issues related to information retrieval methods. With Gordon Bell's personal information as a reference collection, the project started to digitize and store as much information as possible:

“ Since we can't predict when an old bill, conference announcement page, attendee list, or business card will be required, the easiest and safest thing is to simply keep it all.

In the first phase, they scanned all legacy paper documents, photographs and much more. Together with digital artifacts, they put everything into a well-designed folder hierarchy. Phase two was planned with including “real-time capture of conversations, meetings, sensor readings, health monitors, and computer activity”. In the opinion of the authors, with one Gigabyte of non-video data added to the repository, a Terabyte hard disk should be enough to hold eighty years of personal data. They, however, also stated that in the future users “expect to record their lives more extensively”.

The basic idea behind the project, the process of digitizing analog artifacts, and the benefits of having such a complete digital repository is also described in the book “*Total Recall*” (Bell and Gemmell, 2009).

The original project wanted to avoid databases and use folders and [shortcuts](#) instead. However, more powerful abilities were desired during the

project (Bell and Gemmell (2009, pp. 40–46) and Gemmell, Bell, and Lueder (2006, p. 90)).

The new project was called *MyLifeBits*. It basically consisted of a Microsoft SQL server database that held twenty attributes for each item. Depending on the item type, additional attributes could be stored. Links could be made implicit, for example, by time: photos could be linked implicitly with records of GPS positions or a calendar events. Explicit links could be created manually as well.

Legacy applications were also integrated. Microsoft Outlook emails were monitored and NTFS folders were synchronized as well. Many legacy applications could not be integrated, since they relied on hierarchical structure managed within the application domain.

Additionally, the system was able to capture the content of each visited web page, every instant messaging protocol, telephone conversations, radio and TV programs, and all mouse/keyboard activity.

The main interface was the *MyLifeBits* shell, which visualized queries as a list containing thumbnails of images and a timeline view. In contrast to the results, which will be discussed in the Section 3.4.2, “[the authors] deemed search to be the most critical requirement”.

A very clever way of reviewing items was the use of a screensaver. This screensaver application showed stored items at random and allowed annotating items using written text or audio annotation. The authors noted that items like photographs were of better value when they were annotated with explicit metadata as well.

The subjective feedback from Gordon Bell (Gemmell, Bell, and Lueder, 2006) reveals an overall positive picture: “[h]aving a surrogate memory creates a freeing, uplifting, and secure feeling.” He also routinely visited web pages to use the capture mechanism “just to ensure having a copy”.

Discussing the aspect of hierarchical folder structures versus database links the authors noted that “one might conclude that no organization is needed”. In comparison to the traditional hierarchy, they stated that “both views are valid”. Although “[s]pecial skills are required to construct useful classifications”. In relation to the [search](#) versus [navigation](#) topic: “full-text search

2 Research Background

is not enough; in our experience, many items require some other attributes to be found”.

In MyLifeBits, the user had to annotate items, if she wanted to add things not covered by the implicit metadata:

“ But even with convenient classifications and labels ready to apply, we are still asking the user to become a filing clerk – manually annotating every document, email message, photo, or conversation.

To limit this manual effort, the authors answered:

“ [S]top throwing out any potentially useful meta-data. Time is probably the most important attribute in our database.
[...]
[C]apture itself must be more automatic on this scale so the user isn’t forced to interrupt their normal life in order to become their own biographer.

With the development of the Microsoft SenseCam, the surrounding analog world could be digitized a little easier to give impression of what happened. SenseCam was a neck-worn device, which had several sensors for infrared, light, temperature and an accelerometer. By pressing a manual button, each of these sensors were able to invoke a small fish-eye style photograph. This resulted in a series of digital photographs whenever the user moved, stood in front of another person, changed room and so on. Converting those photographs to a video sequence allowed a quick recapitulation of a day or an event. Combining those photographs with GPS sensor data allowed mapping onto a geographic map.¹² Some issues related to data capture with MyLifeBits as well as SenseCam were discussed in Gemmell, Williams, et al. (2004).

Gemmell, Bell, and Lueder (2006) emphasized that “[r]eporting tools with appropriate visualization are very useful applications”. They can give valuable feedback to the user.

12. For a study on the application of SenseCam data, see also Kelly, Byrne, and G. J. Jones (2009).

“ A simple query-based tool can be remarkably insightful and useful from “how I spend my time” to “count and space used” by different items. Reports can track what is being worked on or being thought about, for example by plotting the word “budget” or “nominating committee” against time.

In the last years, the [Quantified Self](#)¹³ movement gained support from many people. Smart mobile devices contain more and more sensors. Dedicated mobile devices for mobile logging of health data also provide useful data.

MyLifeBits was a research platform used mainly by Gordon Bell. To my current knowledge, unfortunately, no screenshots of the search interface were published. It was considered a “proof of concept” (Bell and Gemmell, 2009, p. 50) and thus never released to the public. Aside from subjective feedback, there was no evaluation outside the development team.

2.2.13 Semantic File System

The notion of representing metadata queries as virtual folders was a clever way of achieving backward compatibility in conjunction with new retrieval methods. Gifford et al. (1991) proposed a method for [associative navigation](#) in the form of a semantic [file system](#). They extracted attributes from files and folders using so-called transducers. The authors defined “attribute” as an “field-value pair”. Besides a default one¹⁴, transducers were content type specific. This way, they were able to extract attributes like day, category, subject, author from New York Times articles; subject, recipient, and sender from email messages, and so forth. Using virtual folder techniques, the semantic file system listed matching items by navigating.

For example, the virtual folder `/sfs/owner:/smith/text:/resume` contained items whose owner was the user smith where the word “resume” was included in its content.

13. <http://quantifiedself.com/> – retrieved on 2012-07-31.

14. By default, each item was indexed at least by ownership, group, folder, file name, and [file extension](#).

2 Research Background

Likewise, the folder `/sfs/ext:/mail/from:/bob/subject:/Project` listed items with [file extensions](#) mail whose sender was “Bob” and where the subject was set to “Project”.

The evaluation of the implementation concentrated mostly on performance optimization issues and was basically carried out within the group of developers. Back in 1991, performance was a major drawback of such projects. A set of anecdotal evidence examples showed that the system was “easy to use”. And even “[u]sers outside [the] research group have successfully used the query interface to locate information”. The authors claimed that the semantic file system was effective for information sharing as well as for command level programming.

To my knowledge, the semantic file system presented in Gifford et al. (1991) was never evaluated with test persons or released to the public.

2.2.14 SemFS

A client-server system similar to the one discussed in the previous section was described in Mohan, Raghuraman, and Siromoney (2006). In *SemFS*, metadata, like owner or date of last modification, was used to browse a virtual file system. Additional properties were extracted according to their file format. For example, JPEG files were scanned for EXIF header data or MP3 files were scanned for ID3 data such as length or artist. The UNIX command line sequence “`cd type:mp3^len>3m^artist:Mike`” of a user was interpreted as “`chdir`”¹⁵ into a virtual directory which lists all MP3 files sung by ‘Mike’ of length greater than 3 minutes”. SemFS also provided basic file tagging methods and versioning.

To my knowledge, SemFS was never evaluated in studies or released to the public.

¹⁵. The term “`chdir`” stands for change current working directory.

2.2.15 TagFS

A similar approach to the SemFS system described in the previous section was *TagFS* (Bloehdorn et al., 2006). It consisted of a framework to interpret file system primitives such as “list”, “change directory” as queries in an RDF database and “copy”, “move”, “delete” as change operations on these RDF metadata. This way, a folder *path* resembles a query of a set of tags that fulfill items attached to the tags. Its semantics consist only of a “hasTag” relation, where “adding or removing a file from a directory is mapped to adding or removing metadata”. Hence, “/lisa-ekdahl/favourite translates to `hastag(hastag(/, 'lisa-ekdahl'), 'favourite')`”.

The authors implemented “TagFS as a set of Views and ClassHandlers for SemFS”¹⁶.

It seems to be the case that there is no possibility to delete files within TagFS since the delete command translates solely into a delete command for the metadata associated. Items, whose metadata is removed, are probably no longer reachable by navigating the TagFS. I could not check this assumption for myself.

The source code of TagFS¹⁷ can be downloaded from a web site¹⁸. Due to the fact that the last changes in the repository are dated 2006-07-24, the repository is considered unmaintained. There was no license information attached to the source code.

To my knowledge, TagFS was never evaluated in studies.

2.2.16 hFAD

The position paper of Seltzer and Murphy (2009) discussed the problems of hierarchical file systems and proposed a new approach at a technical level. The authors claimed that the file system hierarchy “has outlasted its

¹⁶. Please note that the “SemFS” of Bloehdorn et al. (2006) is not related to “SemFS” as described in Mohan, Raghuraman, and Siromoney (2006).

¹⁷. The Linux-version of TagFS was implemented in Java using libfuse2.

¹⁸. <http://isweb.uni-koblenz.de/Research> – retrieved on 2012-08-04.

2 Research Background

usefulness”. Seltzer and Murphy argued that “a hierarchical namespace suffers from the following problems”:

- Users no longer know where their files are stored.
- One file might belong to several collections.
- Navigating hierarchical namespaces requires long navigation paths using too many “index traversals”.

Replacement file systems should provide backward compatibility, separate naming from access, treat items as opaque sets of bytes to allow applications to use their own structure, and allow direct access to data. Although databases possess certain properties that would allow for them being a replacement for current file systems, they are “ill-suited” and “heavy-weight” for this purpose.

Seltzer and Murphy (2009) proposed a new architecture for file systems. They called it HFAD which stands for “Hierarchical Filesystems are Dead”. Instead of a hierarchical namespace, HFAD used “a tagged, search-based namespace”. Objects in HFAD had a unique ID and were “named by one or more tag/value pairs”. Additionally to traditional POSIX file operations like read or write, methods like insert and truncate were introduced. They allowed inserting bytes at a given position and removing bytes from the end of an object.

HFAD was implemented with Linux/FUSE, Berkeley DB, and Lucene for full-text indexing.

The authors stated that “the description of HFAD above is incomplete, and there are a variety of open research questions associated with this work”:

“ We encourage other researchers to design their own implementations, and we look forward to the comparisons of different implementations and how well these new system work relative to historical practice.

Although a public source code repository exists¹⁹, it is marked as “pre-alpha”, contains only a source code stub, and was last modified on 2009-07-18. Therefore, the implementation is considered never released to the

19. <http://sourceforge.net/projects/hfad/> – retrieved on 2012-08-05.

public. There is no license information included in the repository stub, nor in the project description.

To my knowledge, HFAD was never evaluated in studies.

2.2.17 Feldspar

The Feldspar²⁰ interface was introduced and described in Chau, Myers, and Faulring (2008b). The specialty of Feldspar was its visually appealing approach of defining associations for information retrieval. By selecting from seven types of characteristics, the user could easily look up “the folder containing the email attachments received through email from Brad or Spence”²¹ or “the webpage mentioned in the email from the person who I met in May”, as illustrated in Figure 2.12.

The interface contained a narrow navigation bar at the top, a large query area in the middle, and a result area on the bottom. While defining a query, the result area already showed results matching the current query defined above.

Feldspar was programmed in C# with Microsoft Expression Blend 2 for its GUI. It used the database of Google Desktop Search as the sole source of information. It supported “seven data types that [the authors] thought were the most common things that people want to find. They [were] *Email, Person, File, Folder, Webpage, Event, and Date*”.

Chau, Myers, and Faulring (2008b) described an evaluation of Feldspar. Eight participants had to pretend to be a fictitious user and use his data²² on his computer to search for information. The within-subjects study compared two conditions: Feldspar and conventional methods of the tools involved²³. Each participant had to do seven tasks (three simple ones, four more difficult ones) on one condition and seven other tasks on the other

20. Feldspar stands for *Finding Elements by Leveraging Diverse Sources of Pertinent Associative Recollection*.

21. Illustrated in Figure 2 of Chau, Myers, and Faulring (2008a).

22. Fictitious emails, files, calendar events and visited web pages.

23. Microsoft Outlook, Google Desktop Search, Windows Explorer.

2 Research Background

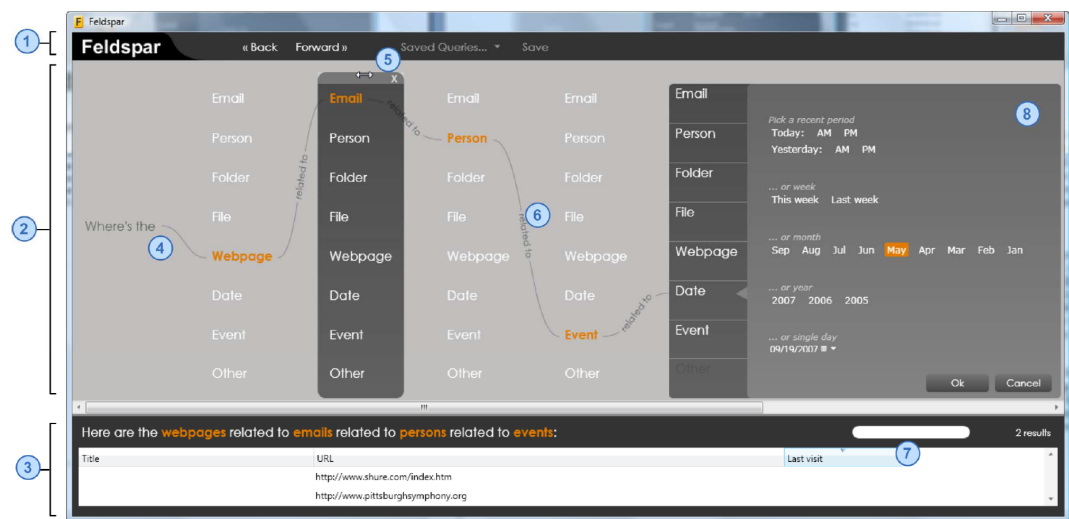


Figure 2.12: The Feldspar user interface. ① The Navigation Bar. ② The Query Area for constructing queries. ③ The Results Area with the query represented as a sentence at the top. ④ The main query area. ⑤ The user can freely edit the type of association and swap its order with other associations. ⑥ Items in queries are linked by the term “related to”. ⑦ The user can filter the results by typing a filtering string into the textbox. ⑧ The date picker panel, which allows the user to pick a data range such as May, or a specific date using a calendar dropdown. At any time, the user can edit the query by selecting different values and the results update immediately. [Source: Chau, Myers, and Faulring (2008b)]

condition. A task was marked as not finished after four minutes without a result.

With the more difficult tasks, participants were significantly faster with Feldspar. The success rate was significantly higher when using Feldspar: only two fails for the Feldspar condition and twenty-four fails for the control group using conventional tools. The subjective feedback (Likert scales) was better for Feldspar in all six categories that were asked.

Chau, Myers, and Faulring (2008a) revealed additional aspects related to design decisions, the Feldspar-predecessor interface *Iolite*, implementation details and future work.

To my knowledge, Feldspar was never released to the public. According to Polo Chau, development of Feldspar was discontinued.

2.2.18 Personal Project Planner, Planz

Like Malone (1983) was a qualitative study on how people organized their desk, W. P. Jones, Phuwanartnurak, et al. (2005) was a study to obtain qualitative feedback on how people organize their projects. It showed that people depended on their folder hierarchy to structure projects. This also emphasized the importance of navigation compared to search. Besides the obvious limitations of folders, the study also revealed some additional issues:

- Aside from alphabetic order, there was not much support for alternative ordering. This resulted in folder names starting with something like “aa_” to enforce items appearing on top of the list.
- Folder names like “images” or “references” were used many times somewhere in the hierarchy of folders. Depending on the context of the folder, such folders were used for organization and re-use of concepts but “scattered throughout [the] file folder hierarchy”.
- Similar sub-structures of file folders could be found for similar projects. However, the traditional folder hierarchy does not support easy re-use of certain structures.

The paper concluded with the following summary of the situation:

“ If, more generally, folders help people to understand and “see” their information better, it is reasonable to ask “can we do better?” What about better representations that make it easier for people to order folders (as they would like to order the components of a project)? Why not better support for the use and re-use of folder structure as a first-class object? (Perhaps we can start by supporting a “Paste Structure” option.). These and other questions naturally arise from the study of how people organize information to get things done.

2 Research Background

In W. P. Jones, Bruce, et al. (2009), qualitative analysis was taken one step further. Twenty-seven participants were interviewed on the projects they were working on. One project was chosen to be representative and follow-up sessions were held, where the participants showed and described the projects to explain the “how” and “why”. Audio recordings of the sessions were transcribed.

The results from this study revealed important insight into how people used tools and methods to organize their projects. Despite the high-technology environment the users were living in, paper was still very important to them. They regarded paper as “satisfying”. Participants liked the idea of having everything in one place. Such project folders containing project files gave them a feeling of being in control. Contact to other people had much impact on the participants. It could be “help or hindrance in a project’s completion”.

“ People can also support a project through direct assistance or by providing information of direct relevance to a project. Of potentially equal importance, participant comments suggest that other people can be an important source of motivation and emotional support.

For example, people may organize their information for reasons similar to those that motivate us to straighten up our houses when guests are coming. We do so as not to look bad in the eyes of others. But we also benefit from the greater order that results.

There was an important aspect, which was is reported by other user studies as well. Participants reported that the fact, that they had to talk about their PIM methods in the study, actually helped them to become more organized.

In order to support users in managing their projects, PIM methods, therefore, should support short notes and snippets, which are independent of the classic office applications. Users should be able to write easily and fast. Having the information in view was an important factor as well: sometimes users need overview, sometimes they want to see tasks which are completed to get a feeling of accomplishment. Similarly to many other stud-

ies, navigation within a structure was recognized as an important method. Folders gave users a feeling of being in control.

W. P. Jones, Klasnja, et al. used the findings from their studies to implement a research prototype, which was called *Personal Project Planner*²⁴(or in short: the Planner). It was described in W. P. Jones, Klasnja, et al. (2008). The Planner was implemented in .NET 2.0 and provided rich-text overlays to folders. User data was stored in XML fragments on a per-folder basis. The GUI traversed the XML files in the current folder and each sub-folder and visualized them in an outline view. References to files, Outlook emails, and Web [Uniform Resource Locators \(URL\)](#) could be easily included at any location:

“ The Planner supports the creation of a rich-text project plan as an external representation that organizes not only component tasks (e.g., “find out what the budget allows”), but also the information needed to complete these tasks. Two principles are key to the Planner’s design:

- No new organization. Project plans are simply an alternate way to view and work with a folder hierarchy in the file system. The headings/subheadings of a plan correspond to folders/subfolders in the file system.
- Organize incidentally. Project-related information is organized as a by-product of a plan’s elaboration.

Many features of the Planner supported easy and fast information input. “Drag & link” was an advanced drag & drop implementation: when a user dragged text from a web page, Outlook message or an electronic document, the Planner also created a link back to the origin of the text. With “in-context create” the user was able to create new documents or Outlook

24. The Planner was influenced by Kaptelinin’s work on a UMEA prototype (Kaptelinin, 2003), Giornata (Vaida and Mynatt, 2009) and WikiFolders (Vaida and Greenberg, 2009). There are some parallels to [personal wiki systems](#) like [Zim](#), [The Brain](#), [Refinder](#) (previously known as Gnowsisi) or online services like [WorkFlowy](#). For the Emacs editor, there is an extension called [Org-mode](#), a very advanced and extensible PIM tool (Dominik, 2010) which can also be used to create highly complex documents (Schulte and Davison, 2011; Delescluse et al., 2011; Schulte, Davison, et al., 2012). The Planner shows the tightest integration into file system folders.

2 Research Background

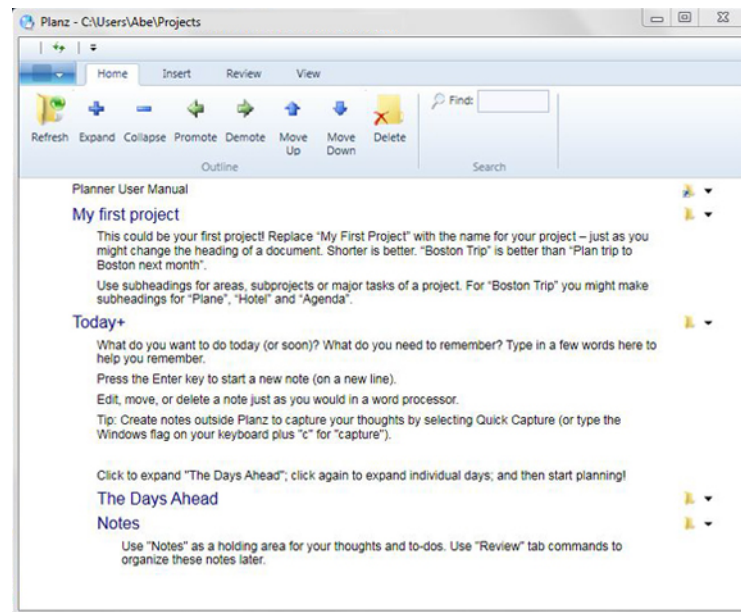


Figure 2.13: Screenshot of Planz (courtesy of William P. Jones).

emails within the Planner. Folders in the file system represented tasks. Re-ordering items was very easy. Outlook information for “remind by” or “due by” could be associated with a folder within the Planner.

The study contained an evaluation of the Planner. It started with a five-minute video presenting the basic features. After installing the Planner on the participants’ computer, an example project was chosen. With this real-world project, the participants had to outline the project’s structure with Planner. Afterwards, the participants had to fill out Likert-scale questionnaires to express their estimated frequency of usage of key features.

Results showed that participants generally rated features like drag & link, system integration, in-context create, outlining and item re-ordering very high. Significantly lower-rated (but also high) were task management and folders as tasks.

In W. P. Jones, Hou, et al. (2010), the Personal Project Planner was renamed *Planz*. Its implementation for the version 7.2 was carried out using .NET 3.0.

2.2 PIM Tools

Planz received several new features: a synchronization with the file system mechanism, HTML export for printing or reusing project data, structure export and re-use, an advanced XML format, and support for additional views.

This study compared Planz to their current work flow for finishing projects. Eight participants were interviewed. For each participant, two ongoing projects were chosen. One was finished with the current tools, the other was finished with the help of Planz. The test users also gave guided tours to explain their current system. Final questionnaires were sent via email. The duration was between five and twelve days with a minimum of five business days.

There was no clear winner in the results. For email management, participants seemed to prefer their current setup with Microsoft Outlook. For getting an overview of the project, participants seemed to have a tendency to prefer Planz.

The most current version of Planz is 8.2, which supports Windows Vista/7, and Microsoft Office 2007. Microsoft Outlook 2007 is recommended. The QuickCapture feature allows quick capturing of items including links. Other new features are better support for flag/check options and in-place folder expansion.

Planz is licensed under a GPL version 2 license and can be downloaded for free on the KFTF-website²⁵.

2.2.19 LiFiDeA and Associative Navigation

Users do not invest much effort to add metadata to information (Rodden and Wood, 2003; Sauermann and Heim, 2008). Yet they approve of using advanced interfaces, which provide retrieval methods based on metadata. J. Kim et al. (2010) described an approach which tried to combine semantics with information retrieval:

25. http://kftf.ischool.washington.edu/planner_index.htm – retrieved on 2012-08-03.

2 Research Background

LiFiDeA show



Figure 2.14: The LiFiDeA user interface. The page of a concept “Search Engine” showing related documents and concepts. [Source: J. Kim et al. (2010)]

“ In an effort to keep the benefit of structured data while minimizing the cost for the user, we propose a simple model of semantic representation for personal information. It is composed of (information) *items*, *tags* and the *links* between items. Items here represent information objects with textual contents. These objects can be *documents* collected from many sources, or *concepts* – entities and terms of interest to the user. Tags and links are the metadata that enables the grouping or the association of individual items.

The paper introduced a prototype system called *LiFiDeA*²⁶ (Figure 2.14). Documents and concepts could be tagged. Information items were collected from several sources including files, email, and [Rich Site Summary \(RSS\)](#) feeds. Metadata was derived during the collection process. Concepts were automatically extracted by the software. The user was able to create concepts manually as well.

Users were able to carry out [faceted search](#). Within the retrieval interface, users were presented a ranked list of related items. Using the tags and

²⁶. The authors derived the name “by combining the words ‘life’ and ‘idea’”.

links generated by the software, users were able to navigate. This provided [associative navigation](#).

When a user clicked on items in the result set, “the system collects the user’s clicks on relevant concepts and uses them as training data for adjusting feature weights”. This method improved the semantic relations of items during system usage.

The authors evaluated LiFiDeA with “two rounds of game-style user studies”:

“ In the first round, users were asked to find the target document using only search and associative browsing between documents. In the second round, concepts were available for searching and browsing, thereby providing a full access to the semantic representation.

Results were quite promising and the system showed a fast learning response:

“ Our evaluation suggests that the semantic representation is useful for known-item finding tasks, especially when concepts are used in addition to documents. Users browsed a lot, and that led to higher success rates. We also found that the combination of features significantly improves the quality of suggestions, and that learning combination weights takes less than 100 clicks.

In J. Kim ([2011](#)) the authors developed a model in order to simulate information retrieval behavior for known item finding.

“ Our goal is to build a reasonable simulation of the actual user behavior for the evaluation of our system for the known-item finding task. The user model is parameterized to simulate various aspects of the user. While we do not argue that experiments based on a simulated user model can be a substitute for user studies, they provide a valuable means of evaluating the system under various conditions, complementing evaluation methods where users are involved.

2 Research Background

The simulated users were expected to start with a keyword search. If the known item was not in a certain range of the result set, users either did another query or started browsing the result set. When a user was not able to locate the item within ten trials, the attempt failed.

User knowledge varied upon their knowledge and ability to make a wise choice within the result set. Another variation was achieved by using either a depth-first or a breadth-first strategy. By altering the value of fan-out, simulated users clicked more or less in the ranked list.

The simulation was run multiple times on a data set of real world information (emails, web pages, publications, lecture notes) in order to consider effects of randomized parameters (user knowledge for example).

In comparison to the study of J. Kim et al. (2010) using human participants, the assumption in the simulated model seemed to be reasonable.

Results showed that “associative browsing is quite effective”:

“ Overall, the informed user with a fan-out of two gave the best performance.

[...]

[O]ne or two clicks are usually sufficient to get to the target document by browsing.

[...]

We can conclude that the user’s level of knowledge has a direct influence on the efficiency of browsing.

[...]

[W]e can infer that higher levels of knowledge makes exploitation more valuable than exploration.

In summary, the simulation experiments show that associative browsing provides an effective (30–40% of success) and efficient (within 1–2 clicks) way of getting to the target document when keyword search is marginally relevant.

LiFiDeA was released to the public on github²⁷.

27. <https://github.com/jykim/lifidea/wiki/> – retrieved on 2012-08-07.

2.2.20 Faceted Search, Faceted Navigation

Information retrieval using facets is a hybrid method of search and navigation. It is widely used in web shops, for example. Figure 2.15 shows a screenshot of the web interface of eBay. Looking for a computer, the user is able to enter a price range, define the device type in general and so forth.

In the example of Figure 2.15, the facets are “Type”, “Screen Size”, “Memory” and so on. Each facet has a set of pre-defined values. Typically, a user can choose no value or up to several values per facet. In the example the values are “Notebook” and “Ultrabook” (for the facet “Type”), “12–12.9 inches” (for the facet “Screen Size”), “2 GB or more” (for the facet “Memory”) and so on.

According to the definition used in Perugini (2009), “Facets are ‘clearly defined’, mutually exclusive, and collectively exhaustive aspects, properties, or characteristics of a class or specific subject”.

The idea of classification using facets is not a new one. They were invented under the name *Colon Classification* of the Indian librarian and mathematician Sirkali Ramamrita Ranganathan (Ranganathan, 1933).

Marti Hearst did important research on modern interfaces using facets (Hearst, 2009, Chapter 8.6). In Tunkelang (2009) all important aspects of facets and their usage are described as well. Tunkelang defines the difference between *faceted search* and *faceted naviga-*

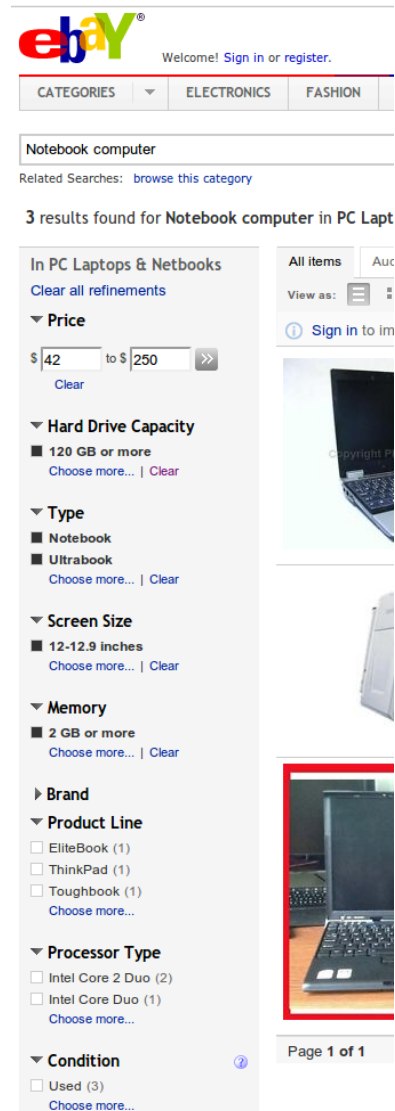


Figure 2.15: Faceted search on eBay.com. [Source: <http://ebay.com> – retrieved on 2012-08-05]

2 Research Background

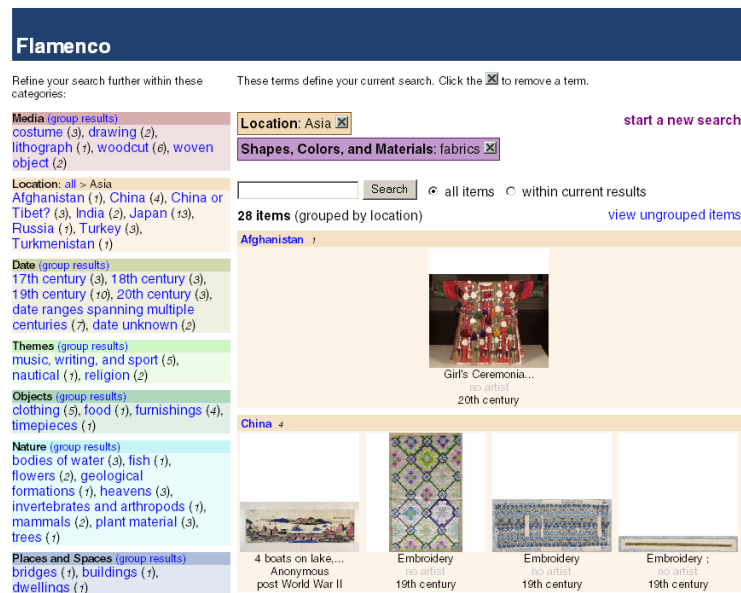


Figure 2.16: The Flamenco faceted search interface. On the left-hand side there are the facets with their values. The currently selected values are “Asia” (facet “Location”) and “fabrics” (facet “Shapes, Colors, and Materials”). [Source: Yee et al. (2003)]

tion such that faceted search is faceted navigation combined with free text search.

There are many information retrieval interfaces which use faceted search or faceted navigation aspects. One of the earlier research projects was *Flamenco* (Hearst (2009, pp. 189–195); Tunke-lang (2009, pp. 33–35)).

Release 1.0²⁸ of Flamenco was made public on the project web site²⁹. It is licensed under a BSD license and was the subject of many studies and papers such as Hearst (2000), Hearst et al. (2002), Yee et al. (2003), or Hearst (2006).

²⁸. It requires Python version 2.3 or higher, a web server such as Apache that runs CGI scripts, and an ODBC-compliant DBMS such as MySQL.

²⁹. <http://flamenco.berkeley.edu/download.html> – retrieved on 2012-08-06.

Tunkelang (2009, Chapters 4–5) lists further faceted search interfaces such as *mSPACE* (mc schraefel, University of Southampton) and *PARALLAX* (Huynh, MIT). *Querium*, a faceted search interface from Golovchinsky and Diriyeh (2011), won the HCIR 2011 Search Challenge at the CHI conference.

Chau, Myers, and Faulring (2008b) elaborated as to why their Feldspar approach (described in Section 2.2.17 and illustrated in Figure 2.12) differs from faceted search:

“ Although faceted search [...] also works incrementally, it is fundamentally different from Feldspar’s multi-level associative retrieval. Faceted search allows the user to find an item by incrementally specifying its characteristics *internal* to the item itself. Multi-level associative retrieval, on the other hand, often requires both *internal* and *external* characteristics. For example, in the query “find the file that was received from *Bob*, authored by Sue, and *modified yesterday*”, *modified yesterday* is an internal characteristic of the file, while *Bob* is an external one. These two retrieval strategies require different algorithms.

I must love what I destroy
And destroy the thing I love

Moon over Bourbon Street
Sting

3 Challenges

This chapter summarizes selected issues, which currently prevent users from having the best possible user experience related to managing their personal data. Subjective conclusions are drawn based on my personal experience and results from cited research work. Some of the challenges are based on the fact that our environment is constantly changing. Other challenges relate to the users suffering from bad PIM. The development of tools either for research or as products has room for improvement as well. Researchers have their “blind spots” by not trying to shift their focus and by not studying issues out of main stream. The methods for obtaining good PIM solutions also affect the output in terms of PIM prototypes or products.

3.1 Changing Environments

The environment we work in is constantly changing. The number of files increases, the applications we use change, cloud-based services store our personal data outside our influence. Future developments will keep on

3 Challenges

challenging the methods and tools we use. What is probably a good solution now does not have to be of any use tomorrow. Things have to be re-evaluated, improved, and constantly adapted.

3.1.1 Increasing Number of Files

The first [file systems](#) were developed in the 1950s. Back then, the focus was to organize and manage, perhaps dozens or hundreds of files due to the clear technical limitations.

With the introduction of hard disk storage built into every desktop computer, a large amount of storage space was available to keep more and more data for direct access. Additionally, the average hard disk capacity doubled every few years. Although the average file size increased as well, this could not compensate the storage space that resulted from this impressive development.

Jarrett and Systems ([1982](#)), still in an office world dominated by classic paper, mentions that the volume of paper used by an average office worker is 20,000 pages and rising at 2,000 per year. The test persons in a study mentioned in Barreau and Nardi ([1995](#)) had 2,400 to 31,000 files on their office computer. But nowadays, an average computer user has hundreds of thousands of files on his primary hard disk, not counting external disks and backups. In Gemmell, Bell, and Lueder ([2006](#)) the personal digital archive of Gordon Bell contained 245,371 items including over 97,000 emails and PDF files of over 70,000 visited web pages. His non-video content grew at approximately 0.5 Gigabytes per month but in a nonlinear fashion. Cho, S. Kim, and Lee ([2009](#)) found that “[t]he average of incremental volume per day was increased by 4.35 times for [user home drives] and 17.42 times for [scratch data drives] between 2006 and 2009.” Boardman and Sasse ([2004](#)) mentioned average growth rates of 100 folders and 1,764 files within an average observation time-frame of 286 days and eight participants.

My personal primary hard disk consumption¹ reflects this trend as well. In Table [3.1](#) it is easy to see that the number of files stored increased

1. A quick informal survey with other people confirmed the approximate quantity and tendency of this data.

3.1 Changing Environments

Month	Capacity [GB]	# Files	# Extensions	# Folders	Files/Folder
1999-06	20	3135	150	225	13.9
2001-02	80	15599	377	1322	11.8
2012-06	2048	732296	3092	73004	10.0

Table 3.1: The statistics of my personal file system shows an increasing amount of information. Each row consists of a month where the snapshot was done, the size in gigabytes of the primary storage device that was used, the number of files, the number of file extensions (reflecting the file type variance), the number of folders, and the average ratio of files per folder.

dramatically up to over 730,000 files. This means an increase over three-hundredfold within thirteen years.

Not only the number of files exploded: the increase of the number of different [file extensions](#) demonstrates a much larger set of different file types to manage: from 150 different extensions² in the year 1999 up to over 3000 different file extensions in 2012.

To organize this amount of data, the number of folders scaled up as well. It is noteworthy that the average number of files per folders stayed almost the same, not changing in magnitude at all. Cho, S. Kim, and Lee (2009) found similar numbers with an average of 16.85 files per folder within users home folders.

With the number of files increasing and a more or less stable files per folder ratio, files might tend to be stored more deeply in the hierarchy of folders. This assumption was confirmed by Cho, S. Kim, and Lee (2009) as well:

“Most of the directories have less than 10 directory depth. We detected that the averages of directory depths were increased by 1.16 in home directory and by 3.22 in scratch directory between 2006 and 2009.

So it is the case that many files had to move into folders which are deeper in the hierarchy of folders compared to several years ago. This means that

2. The number of extensions normalized to lower case characters and filtered for file extensions containing only combinations of characters “a” to “z” and numbers.

3 Challenges

our navigational processes must have changed to visit more folders than in the past.

A very high percentage of files never gets accessed at all. Observing large sets of data for 157 to 287 days on different computer systems, Gibson, Miller, and Long (1998) noticed that “fewer than 30% of the files [...] were used”. Over a three month period, Leung et al. (2008) detected no access for “more than 90% of the active storage”.

This high percentage of never-accessed files could indicate users’ inability to locate them or the lack of knowledge of existence of those files. Strict folder hierarchies do not support serendipity very well.

Following the comment from Lansdale (1988, pp. 57, 59), there might be a high chance that a dramatic change in the environment (number of files to organize, files deeper down in hierarchy, many files never accessed) combined with not changing the method (file system with strict hierarchy of folders) probably leads to a system which does not fulfill the requirements anymore.

3.1.2 Non-Textual Content

In the last century, most personal information was based on text – either in human-readable text files or in pseudo-standardized binary formats for office products. As shown in the previous section, not only has the number of files increased since then. The explosion of the number of file extensions is a sign of a dramatic increase in file formats. In Chapter 12 of Hearst (2009) these “emerging trends” towards multimedia content are described. Much information is stored within audio, video, and photographs. Conventional information management tools are not able to search this kind of content or extract valuable key phrases. Different retrieval techniques have to be developed (Hearst, 2011), otherwise, a blind spot in PIM will emerge.

3.1.3 Scattered Storage Locations

A few years ago, the personal information of a user was more or less on one single device: her desktop computer. Today, many users possess a wide range of devices, each holding a valuable fraction of their personal information: a desktop computer at work, a desktop computer at home, a notebook, a smart-phone, miscellaneous hard disk storage devices for extension and backup, and probably one of those nifty new tablet computers. This information storage mess is made even worse by a large number of different cloud services storing personal information. We simply can not tell any longer, where our data resides. This information fragmentation is discussed for example in W. P. Jones (2007, p. 392), W. P. Jones and Teevan (2007, p. 270), and W. P. Jones (2012).

One possible solution to this problem of fragmentation is unifying (W. P. Jones and Teevan, 2007, Chapter 8). This could be done at many different levels. For example, we can try to bring our data back to a single place. This allows for *central* indexing, navigating, and using our data. Such an approach was chosen, for example, in the MyLifeBits project (see Section 2.2.12).

I created a similar open, extensible solution called *Memacs*³. It integrates my personal data from sources like text messages (phone), phone calls, photographs taken, updates on Twitter⁴, rss feeds, bank statements, delicious-bookmarks⁵, version control commits (svn and git), email archives (mbox or IMAP), Google calendar events, newsgroup postings, and so forth back to my computer and onto my calendar. This way, my personal calendar turns into a very handy tool which allows me to recapitulate what happened when in which context. Additionally, this collected data is an important source for looking up information related to me. Some data might still be located somewhere in the cloud, but with Memacs, I am able to locate it on my system and follow its link to the original data source.

3. Memacs is based on Emacs and Org-mode <http://orgmode.org> – retrieved on 2012-08-10. The term “Memacs” is a combination of “Emacs” and “Memex”. It is described and published on <https://github.com/novoid/Memacs> – retrieved on 2012-08-10

4. <http://twitter.com/n0v0id> – retrieved on 2012-08-10.

5. <http://delicious.com/vk/> – retrieved on 2012-08-10.

3 Challenges

3.2 Users

Persons using PIM tools are the focus group of PIM research. However, this group of people is neither homogeneous, nor static. Novice young users, eager to play around with everything they can grab. Novice elderly users who need assistance with exploring possibilities and features. Users who need to use interfaces outside of their usual expertise. Experts who need to have shortcuts and optimize their methods as much as possible. And those are only a few examples of a whole spectrum of users out there. In many cases, they all want to use the very same tool, the same interface. This is a difficult task to do right.

For some challenges, we can develop methods and features to support this variety of use cases. However, there are many challenges where the users are able to improve their workflows enormously by learning and applying simple things. Everybody in this whole system, from researchers to developers and users, is able to contribute to the situation.

3.2.1 Time Spent While Retrieving Information

Getting the right information at the right time is a crucial factor in our information-workers' world. There are many studies showing that a large fraction of our daily work is lost due to inefficient search for information: Feldman and Sherman (2001), Morville and Rosenfeld (2006, pp. 11–12), Baeza-Yates and Ribeiro-Neto (2011, pp. 642–643), W. P. Jones (2007, p. 218), Accenture survey 2007⁶, and many more. A current study by McKinsey (Chui et al., 2012) reported “28 hours each week is spent by knowledge workers writing e-mails, searching for information, and collaborating internally”. The same study came to the conclusion that there is “20–25 percent potential improvement possible in knowledge worker productivity”. Hence, up to *a quarter* of productivity is wasted nowadays.

6. <http://www.businesswire.com/news/home/20070104005159/en/Managers-Majority-Information-Obtained-Work-Useless-Accenture> – retrieved on 2012-07-18.

This problem is getting worse over time, as the findings in Section 3.1.1 indicate: we have to manage more and more files, we have to locate them deeper in the hierarchy of folders, and we take longer navigational paths reach to the files.

Large amounts of time and money could be saved if at least some of these issues could be resolved. Technology can not guarantee to re-gain all of this lost productivity, but many studies mentioned in Chapter 2 prove that there is room for improvement in PIM.

3.2.2 The Need for Diverse Interfaces

As stated above: users are different. Each and every one. Then how can we address this diversity in designing user interfaces for PIM?

A promising approach is the customization of user interfaces. In a very basic form this means that any tool provides a reduced set of features and a simplified version of its interface *by default*. An advanced version of its interface could be activated by expert users at any time. More elaborate software tools could allow a fine-grained set of feature-enabling preferences.

Reduced GUI design is one of the key aspects of the popularity of Apple's os x or Google's web services. In the western culture⁷ this is also perceived as clean, easy to use, and inviting to use and explore.

Contradicting the assumption of most software developers, the average user only needs a small subset of features and GUI elements visible. Everything else is clutter, taking away the user's attention from the important things. It is not an easy task to define which feature is essential and which is not. Some tools, like popular web browsers provide only a basic feature set, and can be extended by a large number of add-ons or extensions available for download.

7. According to Reinecke and Bernstein (2011), GUI interface design is perceived very differently by persons having a different cultural background. For example, a crowded interface design is favored in other countries.

3 Challenges

Such diverse interfaces scale well with the growing knowledge and changing requirements of their users. For interface design, “one size fits all” is always a trade-off, only really fitting a certain percentage. Users should be able to customize the tools they use.

3.2.3 Teaching and Learning PIM

The assumption that most users are deliberately trying to optimize their daily workflows is misleading. As Tognazzini (2011) mentioned for engineers, it might be a common misconception that researchers are average users. We are not. In fact, PIM researchers are probably the opposite of the average user. Studies like Lane et al. (2005) or Peres et al. (2004) show that on average people do not use even basic keyboard shortcuts like Ctrl-C/Ctrl-V (for copy&paste), Ctrl-S (save), or Ctrl-F (find). Despite the fact that using such keyboard shortcuts optimizes our level of efficiency, they are seldom used.

However, keyboard shortcuts are not the whole story. Managing a huge amount of personal information on multiple devices requires much more skill than just being able to copy and paste using the keyboard. When a new method is available, people seem to adapt to the new possibilities slowly. Sauermann and Heim (2008) did a study with tagging data and remarked that “[g]enerally, participants needed some time to learn how to model effectively”. Participants do not have a cognitive model for their tags right away. It needs time and experience to find a suitable way of making use of tools providing tagging methods. Ulises Mejias summarized best practices in an essay promoting *tag literacy*⁸. In order to get the best out of tagging systems, users have to learn these things either from others or through their own (bad) experiences.

In my opinion, users which are new to tagging tend to start as *describers*, using many tags which are also directly contained in the item itself. With more experience on re-finding items using this new method, users tend to become *categorizers* (Körner et al., 2010). Future long-term studies will

8. <http://blog.ulisesmejias.com/2005/04/26/tag-literacy/> – retrieved on 2012-08-12.

bring more light to this topic. Anecdotal evidence shows that a **controlled vocabulary** (cv) is often seen as an unnecessary limitation at first. Therefore, this feature is seldom used by novice users starting to tag items. People start to get a feeling on the positive aspects a cv can provide either through getting in touch with an experienced person who uses a cv or by learning from bad experiences with homonyms, synonyms, singular, and plural. However, most users never get this far. Best practices are not available or read only by a minority. The group of “non-changers” from Boardman and Sasse (2004) comprised the majority of participants in their study. Even when people are directly confronted with thoughts about their PIM workflow, most people do not seem to change their behavior at all. Haraty et al. (2012) described issues regarding feature adoption by users.

There are exceptions. In both, Boardman and Sasse (2004) and W. P. Jones, Bruce, et al. (2009), the authors mention strong influence on some participants only because of the fact that they were part of a PIM study. Some people sensitive to this topic seem to get enthusiastic about optimizing their PIM workflows when they are forced to think about it.

Why not familiarize such users with thoughts on PIM in the first place? Being a participant in a study should not be the only chance to raise this issue.

School The usual place where people learn basic knowledge is within the educational system. Unfortunately, PIM methods are not part of general school education. It seems to be the case that each and every one has to develop PIM methods alone in practice – without having the opportunity to benefit from best practices that have already proven to be of use. Computer science⁹ courses in most European schools consist of the ECDL¹⁰. The structure and content of the ECDL is aligned with Microsoft Office Products. Unfortunately, it is more about the application of Microsoft Office features

9. In some countries this class is referred to as informatics or similar. In the UK it is called ICT (Information and Communication Technologies). There is now a move away from teaching ICT (Microsoft applications) in school, toward teaching computer science (including programming and so forth).

10. <http://www.ecdl.org/> – retrieved on 2012-08-09.

3 Challenges

and less about teaching concepts or different products at all. In my opinion, such topics should be moved to other classes. For example, working with spreadsheets should be learned in scientific subjects like mathematics or physics. In computer science class, pupils should learn basic digital literacy and thinking about concepts and algorithms instead.

Books If a person is sensitive to topics like PIM, she is able to get in touch with other peoples' summarized knowledge by reading books. Books are still very popular. And there are many good books about PIM as well. *Bit Literacy: Productivity in the Age of Information and E-mail Overload* (Hurst, 2007) covers basic issues of PIM from a pragmatic point of view. It is full of best practices and tips¹¹ for common workflows. For example, it explains why choosing the JPEG format for screenshot images is not a wise choice at all. I think the things mentioned in the book by Mark Hurst should be common knowledge for *everybody*. To some extent, the **Getting Things Done (GTD)** method described in the best-selling book *Getting Things Done – The Art of Stress-Free Productivity* (Allen, 2001) is one of the most popular examples on how PIM could actually inspire a large group of people. David Allen's company is very successful giving talks and mentoring managers to optimize their PIM workflows. A very thorough book on PIM is *Keeping Found Things Found: The Study and Practice of Personal Information Management* by William Jones. Unlike typical specialist literature, it also addresses the average user. Although it is well backed up with many important references to scientific literature, it is definitely recommendable to non-researchers looking for answers.

Web As with other subjects, the world wide web is an overwhelming source of information also on PIM. *Inbox Zero*¹² is a method to cope with email overload¹³. As part of the Keeping Found Things Found project, *Tales*

11. Chapter 5 on managing todos advertises a commercial web service for managing tasks using email. I would have proposed something independent. I totally agree with almost anything else in this book.

12. <http://inboxzero.com/> – retrieved on 2012-08-09.

13. The common problem of email overload is described in Whittaker and Sidner (1996) and many other studies and books.

3.3 Tools and Development

of PIM¹⁴ started a discussion on PIM topics in a forum. Once sensitized, users really like to share their experiences on their own PIM. Many blogs relate to this topic and movements like [Quantified Self](http://quantifiedself.com/)¹⁵ are more or less part of PIM. And there will be much more to come in the future.

People As mentioned above, many people like to share their experiences with others. Peres et al. (2004) showed that social factors are the most important aspect when it comes to learning how to use a computer efficiently. Working on the same computer with another user who is able to do things more efficiently, is a very good method to propagate PIM knowledge. Almost everybody knows a co-worker in the office who is the expert to ask. The same holds for PIM methods.

It seems to be the case that awareness for PIM related issues still has to be created. As mentioned in Section 3.2.1, there is a huge amount of lost productivity which could be reduced.

3.3 Tools and Development

Missing knowledge or techniques are not always the reason for suboptimal PIM tools. There is this old problem of “reinventing the wheel”. Many good ideas or small notions were part of research projects that were discontinued or just not well referenced. Already developed progress has been forgotten. Or, perhaps, the research software never left its prototype status, not allowing its potential to be developed.

3.3.1 Missing Availability of Advanced Management Solutions

Lansdale (1988, p.63) discusses how to get good solutions for managing information:

14. <http://talesofpim.org/> – retrieved on 2012-08-09.

15. <http://quantifiedself.com/> – retrieved on 2012-08-09.

3 Challenges

“ How then do we proceed to something different and more functional? *It must be by an iterative process* rather than purely by the adoption of some new scheme or technology. It is true that new possibilities of the technology will be recognised: some of these, *by natural selection*, will succeed and some will not. These will lead to methods which the user will have available for all aspects of his job, including information management. From the other side, clear user needs (such as reminders, for example) will emerge from behavioural studies of office activities for which suitable interface devices will be envisaged. Again, some of these will succeed and some will not. Thus far, this is how the development has been driven; by technological development and by the automation of existing strategies. *[Emphasis by Karl Voit]*.

However, in the year 1988 when those lines were written, a completely different ecosystem existed. Many different companies were competing for a diverse market with no clear leader and no clear standards. A variety of operating systems, each with several mostly data-incompatible applications were struggling to find their customers.

Since then, the computer landscape has changed dramatically: the Microsoft Windows platform gained dominance and Microsoft Office software products form a some kind of industry-wide standard. For almost all requirements, there are few market leaders with “standard software” products. The feature-sets of those products define the set of possibilities for their customers. If a market-leader does not implement a certain method or feature in its software product, this method or feature can not be used by the huge majority of users.

This is not a “survival of the fittest” situation at all.

Even worse, a market leader having a large share of users has little incentive to change products in a more than an iterative way. *Backward-compatibility is more important than innovation.*

Where does innovation come from instead? Academic research has the potential to develop methods and solutions which do not necessarily promise

3.3 Tools and Development

financial profit. To a certain degree, these research software prototypes do not even have to be compatible with existing applications. Chapter 2 lists many research projects that developed impressive and innovative things. Unfortunately, the majority of these solutions (fifteen) were never published for public download and usage. And of the few that went public at all, three are discontinued/outdated and one is available only as a binary. After a research project has ended, source code and knowledge often becomes lost.

Developing software products for the masses requires a different kind of development process and maintenance than developing research prototypes that just have to run long enough to return research results.

The open source software model has come far. Software like the Linux kernel, the GNU project, the Samba server, the Apache project, and so forth are modern, robust, and became the driving force behind the Internet.¹⁶ Not only free “versions” of commercial software were developed by this community. It is also a playground for many innovative projects. One promising project is Nepomuk¹⁷, the Semantic Desktop in KDE. Among the many users of the KDE desktop, only a small minority embraces the advanced possibilities of having software tools for managing data in a semantically enriched way.

Given the fact that there exists a new software product that implements a new and innovative method for information management that helps users, there are other circumstances that lead to user acceptance. In Voit, Andrews, and Slany (2009) the authors summarize a set of issues that are often neglected and lead to rejection. There are many ways to do it wrong and so few to do it right.

The basic need of users to enhance their PIM practice is unquestioned, as the quote on page 18 confirms.

¹⁶. Most email servers, web servers and cloud based services use open source software.

¹⁷. <http://nepomuk.kde.org/> – retrieved on 2012-07-18.

3 Challenges

3.3.2 Degree of Maturity of (Tagging) Interfaces

I have seen many studies that did not seem to recognize the fact that comparative studies should compare tools which have a similar level of maturity.

For example, tagging interfaces rely very much on the way they are implemented. The purpose of a tagging interface is not only to provide the ability to add tags to items. Given that the user of such an interface has to invest effort to add metadata, the interface has to make sure that this effort is kept to an absolute minimum.

Completion of already known tags and a tag recommending system are very basic features supporting the process of adding metadata. A tagging interface without [tag completion](#) is like a file browser where the user has to actually *type in* the name of each folder to open.

A number of comparative studies, where the level of maturity was not comparable, is discussed in detail in Voit, Andrews, and Slany ([2012a](#)).

3.4 Research Topics

In the last decade, the focus of many studies and researchers was on search-based information retrieval. The increasing performance of average desktop computers allowed reasonable use of desktop search engines. It was tempting to believe that search will replace navigation in PIM as long as the search engine showed optimized result sets.

If such a perfect search engine existed, many problems related to PIM would be gone without changing much of the rest of current systems. All known disadvantages of the strict hierarchy of folders, of dead-end navigation processes, of the huge amount of items to manage could be ignored: the perfect search engine would solve recall problems independently of the rest.

Unfortunately, this is not the case.

We have to also consider changes in major parts of current systems in order to enhance PIM more. This is about choosing a research focus.

3.4.1 Desktop Metaphor as a Limiting Factor

The desktop metaphor was an ingenious trick in the 1970s. As described on page 1, it was a clever way to transfer knowledge from a real world scenario to the new virtual world of computer interfaces. With the desktop metaphor it was possible for non-tech-savvy persons to use this kind of technology. This metaphor has a clear advantage in terms of learning.

Meanwhile, the world has changed much. In many countries, younger generations have grown up with technology. A computer is not an uncommon thing anymore. Interacting with basic user interface elements is wide-spread knowledge. Many children are more capable of using modern technology than their parents.

The desktop metaphor is still around, but its ability to teach us how to use a computer is not necessary any more. For example, web pages do not follow this metaphor at all. Hyperlinks offer access to information from an arbitrary number of pages. Web-based shops present a single item in *all relevant* categories, not just one.

Since the desktop metaphor is based on real world experience, one key element is that *one thing has its one location*. Our file systems were designed according to this scheme: one file has exactly one storage location within the file system.

Consider the following scenario: a user has organized her own files in a hierarchy of folders where one sub-hierarchy consists of one folder for each contact she knows. Another sub-hierarchy holds one folder for each project she is currently working on. She now receives a file with her email system which holds information about a co-worker Bob's idea about a specific project. This situation is illustrated in Figure 3.1.

On the one hand, it is perfectly reasonable to file this information in the folder related to Bob since he is the author of this file. On the other hand it is also perfectly reasonable to file this information in the folder related

3 Challenges

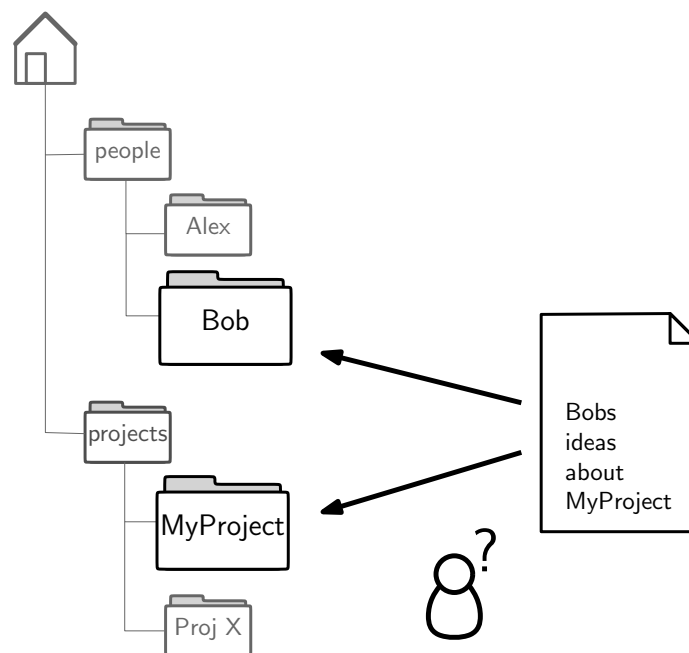


Figure 3.1: An important problem of current file management: strict hierarchies of folders allow only one destination folder, files can not be stored in more than one folder. The user has to apply workarounds: (a) put the file in one folder and use links in other folders, (b) put copies of the file in multiple folders, or (c) maintain an implicit prioritized cognitive model of the folder hierarchy.

to the project since its content is about the project itself. Unfortunately, modern file systems do not allow one file being in two folders at the same time.

People around the world are facing this problem multiple times a day. We all use common workarounds to cope with this dilemma:

- creating links to the file in each folder that is also relevant, or
- storing copies of the file in each relevant folder, or
- deciding, which folder is most suitable and sticking to a single destination folder.

Links The problem with links is that most implementations are not designed in a way that the link keeps working when the original item is moved or renamed. Those broken or dangling links result in disappointing user experiences. Another issue with links is that users simply do not use them (Gonçalves and Jorge, 2003). This is probably because users are not aware of being able to link files. Or, perhaps, users are disappointed because of broken links in their previous experience.

Multiple copies Users filing information into any folder which may be relevant, results in wasted storage space. Each additional copy requires as much storage capacity as the original file. The problem of inconsistency is even more severe. When the user modifies one copy the others do not get modified as well. The content of the files diverges and the user might forget which copy is the most current. On shared network drives and hard disks of employees, studies¹⁸ estimate the percentage of avoidable storage space due to duplication of information at roughly 10–30 percent of the total storage space. This is a huge amount of wasted capacity. The negative side effects from inconsistent updates result in additional hidden costs due to inefficient work and the use of outdated information. There are many business models that try to ameliorate this problem. The current state of research in deduplication methods is described by Malhotra, Sarode, and Kamble (2012).

Cognitive model Most users decide which folder to choose using the priorities of their implicit cognitive model. Some users might think that the project folder is more important than the alternative folders. Some users might prefer the project folder of the contact. Those decisions are made having a certain state of the cognitive model and within a specific situation the person is in. When the user wants to re-find the information, her cognitive model might have changed in the meantime. And her situation while accessing the information most likely changed as well. Sometimes we won-

18. <http://hubdesigns.files.wordpress.com/2010/04/hidden-costs-of-duplicate-customer-data.pdf> – retrieved on 2012-07-18.

3 Challenges



Figure 3.2: A humorous summary of the impossibility of finding *the* right organization method (ALT-text: "Lifehacking!"). [Source: <http://xkcd.com/1077/> – retrieved on 2012-08-20].

der why we filed an information in this or that folder¹⁹. This demonstrates the drawback of this kind of "filing workaround": re-finding is difficult when our models and situations change. The "mental map" of our folder hierarchy keeps changing, but the files stay in the same folders where they were originally filed.

In the example with the filing dilemma mentioned above, the desktop metaphor is actually a limiting factor that results in poor system support to the user. In this case, the desktop metaphor is similar to crutches: an essential need for the technologically handicapped, but a handicap for the technologically healthy.

A computer interface without these limitations could assist the user in many possible ways. The virtual computer world could benefit from allowing *each thing its places* instead of each thing at a single place²⁰. We should think about alternative interfaces that ignore the limitations of the

¹⁹. Such incidents were mentioned, for example, in Dumais et al. (2003, p. 78) or Boardman and Sasse (2004, p. 587)

²⁰. This is also a fundamental idea in Weinberger (2007).

desktop metaphor in order to provide better interfaces which assist users in their routine work.

In Barreau and Nardi (1995) the authors presented two studies where they took a closer look at how users organized information on different operating systems²¹. By observing users, interviewing them and asking them to give a tour of their system, the authors found out several things:

“ [W]e found that *users prefer filing by location* because it aids in helping them find what they need as well as serving a crucial reminding function. We found that *users do not expend great energy on archiving* because old information is generally not useful information. We found that *users give up on elaborate filing systems* because in the end they do not yield enough value. *Users file information not according to systems of keywords or carefully architected logical schemes*, but according to the dictates and vagaries of the kind of work they are doing and the type of information they are dealing with. [Emphasis by Karl Voit].

Fertig, Freeman, and Gelernter (1996) came to the conclusion that the desktop metaphor “favors certain types of interaction over others” and responded to the paper from the previous paragraph:

“ [W]e believe that Barreau and Nardi’s findings are mostly artifacts of the desktop and file&folder metaphor. The desktop metaphor was created on analogy to our paper-based world. *Our computer-based systems can do better*. There are many emerging systems that go beyond our traditional file systems and user interfaces that are ripe for study. [Emphasis by Karl Voit].

Many other research papers emphasize the fact that the desktop metaphor should be considered obsolete: Lansdale (1988) mentioned that “[i]t is important to recognise that the design philosophy of these [desktop metaphor based] machines is not principally to support information management, but to make the machines easy to use.” And: “in areas of information management which involve longer term storage and retrieval, their facilities do

²¹. The operating systems were Microsoft dos, Microsoft Windows, IBM os/2, and Apple Macintosh.

3 Challenges

not provide added value.” Perugini (2009) emphasized the fact that accessing information over a variety of paths is an important notion. Houben (2011) took a closer look at a post-desktop-metaphor world from the perspective of activity theory. Indeed, numerous books and studies came to the conclusion that the desktop metaphor, with its restricting properties, should be subject of change: Nielsen (1996); Shirky (2005); Weinberger (2007); Seltzer and Murphy (2009); and many others.

It is now obvious that the advantages of the desktop metaphor no longer dominate its negative effects. The desktop metaphor is a relict of accustomed limitations from the physical world. It has well earned a life in retirement. *Let’s focus on new possibilities.*

3.4.2 Neglecting the Importance of Navigation

Research in information retrieval has focused more or less only on web navigation or [search](#). Only a small fraction of studies relates to local folder [navigation](#). Yet, this is exactly what many millions of users are doing many times a day: looking up files and folders on their computer. Even worse, this process is not optimized at all.

Many studies provide impressive evidence that, for users, navigation is more important than search for (local) information retrieval. It is worth noting that most of those studies were trying to test and improve *search*, not navigation.

“ Browsing was a strategy used frequently to retrieve files. It was the observed method of choice among the subjects when looking for old files, but it was used just as frequently to retrieve documents from a current list. (Barreau, 1995)

“ In any case, our participants most commonly wanted to browse their personal photos by event, rather than querying them based on more specific properties. (Rodden and Wood, 2003)

“ [W]e observed that even when people could use their contextual information to teleport directly to their information target, they often preferred to orienteer to the information instead. [...]

These possible explanations suggest that future systems should deeply consider orienteering or other approaches to help people use contextual data, perhaps by prompting them with contextual information instead of requiring them to fully specify all of this information at query time. (Alvarado et al., 2003)

“ Participants reported a strong preference for browsing over search in all three tools. (Boardman and Sasse, 2004, p. 587)

“ We found that often keyword-based search engines were not used when searching, and when they were used, it was usually part of an orienteering strategy. The observed advantages of orienteering include the fact that orienteering allowed participants to not fully specify their information need up front and enabled them to take advantage of the large amount of contextual information they knew about their information target. We have suggested that search tools should support this orienteering behavior. (Teevan et al., 2004)

“ The standard method of document retrieval from personal work space for these managers is the same as that used 10 years ago: by browsing an ordered list and selecting the desired item. [...]

All of the managers use search engines such as Google to find things in shared spaces, including the Internet, but they rarely employ a search engine to find information in their personal work space. One of the 4 said that he occasionally uses the find feature within his e-mail software to separate a group of messages from a person or on a topic, but not often. (Barreau, 2008)

3 Challenges

“ Our results show preference for navigation over search regardless of the use of improved desktop search engines.
[...]

As navigation is the preferred retrieval method used by millions of users many times a day, it needs more research and development than is currently focused on it. Shortening navigation time, even by a second, could accumulate to several working years per day for the entire population. (Bergman et al., 2008)

The need for research work related to navigation is omnipresent in many research results. While search is an important part of our PIM portfolio, navigation is used much more often and its tools did not evolve to the same extent in the last fifty years.

3.4.3 Local File Management Not Elaborate Enough

The previous section described the importance of navigation. This section emphasizes the fact that the current state of local file system navigation is not elaborate enough, leaving much room for improvement. A small selection of strategies and approaches to overcome this limitation is mentioned.

The most important drawback of strict hierarchies is probably demonstrated by Furnas, Landauer, et al. (1987). They studied the *Vocabulary Problem* by letting people name things and compare their choices. The results showed that “[i]n every case two people favored the same term with probability < 0.20 ”. Their conclusion was that “[m]any, many alternative access words are needed for users to get what they want from large and complex systems”.

Although this study was about finding terms where *different* users agree upon, it is likely the case, that one single user might face similar problems when there is a large time gap between naming an item and re-finding it. If this holds true, a user faces a huge problem when using a hierarchical structure where an item has only one single location and a single name. Lansdale (1988) confirmed this assumption:

“ The recall process is deficient because, as we have seen, the users’ ability to categorise documents with the appropriate filenames, and their ability to remember those filenames, is limited.

Section 2.1.3 summarized the work of Lansdale (1988) where the author made the point, that a strict hierarchy of categories has many issues.

Shirky (2005) wrote a very good essay about the need of a paradigm shift away from single classification ontologies. Using great examples, he made his point very clear. In relation to file systems and hierarchy he wrote:

“ Browse versus search is a radical increase in the trust we put in link infrastructure, and in the degree of power derived from that link structure. Browse says the people making the ontology, the people doing the categorization, have the responsibility to organize the world in advance. Given this requirement, the views of the catalogers necessarily override the user’s needs and the user’s view of the world. If you want something that hasn’t been categorized in the way you think about it, you’re out of luck.

The search paradigm says the reverse. It says nobody gets to tell you in advance what it is you need. Search says that, at the moment that you are looking for it, we will do our best to service it based on this link structure, because we believe we can build a world where we don’t need the hierarchy to coexist with the link structure.

Shirky, however, ignores the fact that, for personal file systems, there might be a valid alternative representation which supports navigation. This alternative system must not be static. Simplifying the problem to the point that only search is the answer does not cover the whole story.

Similar ideas are described in the book *Everything Is Miscellaneous: The Power of the New Digital Disorder* (Weinberger, 2007). Weinberger analyzes the problem of single structures and its background in history. He proposes the “third order of order” which basically means no fixed order at all. In fact, with *digital* information, the order of representation should be derived upon access, with regard to the requirements of the information retrieval

3 Challenges

purpose. The content is indexed and used as metadata for querying as well. Organization is deferred until the point of use.

Orlowski (2002) describes some aspects the developers of the BeOS file system were trying to solve. They designed a strict hierarchy of folders as only one representation of file system content. Multiple representations of the same set of stored items could be generated.

Rodden and Leggett (2010) described an interesting approach of Google Gmail to combine aspects of search, tags (labels), and folders in the same interface. These methods allowed multi-classification for users who care about it and conventional folders for users who wanted to keep on using the well-known hierarchy metaphor. Another product from Google, *Google Docs*²² also allowed enhanced multi-classification. However, with the introduction of *Google Drive*²³, the feature of assigning multiple labels to items was removed again²⁴.

Chapter 2 contains descriptions of many research tools that implemented a variety of possible solutions, including the following:

Temporal Views Visualizing items in a temporal view was implemented in tools like Lifestreams (Section 2.2.3), TimeScape (Section 2.2.4), or SIS (Section 2.2.9).

Links The use of hyperlinks and associative paths is a central idea behind tools like Presto (Section 2.2.6), Haystack (Section 2.2.7), Planz (Section 2.2.18), and LiFiDeA (Section 2.2.19).

22. <http://docs.google.com>, was replaced by its successor Google Drive <https://drive.google.com> in mid 2012.

23. Google Drive resembles an online data storage like the popular Dropbox service <http://dropbox.com> – retrieved on 2012-08-10.

24. I personally think the reason was that Google Drive was extended with direct synchronization to users' desktop computers. And those desktop computers are using traditional strict hierarchical file systems.

3.5 Research Methods

Unified Hierarchies Boardman tried to at least provide a user experience consistent with strict hierarchies by unifying them with WorkspaceMirror (Section 2.2.8).

Facets A promising method to enhance interfaces for re-finding are interfaces using facets. Some approaches mentioned were Phlat (Section 2.2.10), Feldspar (Section 2.2.17) (at least to some extent), Flamenco, mSPACE, and Querium (all in Section 2.2.20).

Semantics Although not part of common computer desktops yet, semantics have a great potential. With new file systems, the Semantic File System (Section 2.2.13), SemFS (Section 2.2.14), TagFS (Section 2.2.15), and hFAD (Section 2.2.16) all provided a new way of managing files. The Nepomuk²⁵ project added semantics to the KDE²⁶ desktop, a popular GNU/Linux desktop environment. Many researchers are working in the field of semantics which might be applied to file systems (Faubel and Kuschel, 2008). Combining navigation and semantics is also an interesting topic (Solskinnsbakk and Gulla, 2010; Helic et al., 2010; Helic et al., 2011; J. Kim et al., 2010).

3.5 Research Methods

Researchers do not only choose appropriate research topics. They also choose research methods to investigate and evaluate their findings. Research methodology is a very sensitive subject. Certain aspects are important influence factors for the outcome. This section lists two examples, where I believe that a change of mind might bring better research results for PIM.

25. <http://nepomuk.kde.org/> – retrieved on 2012-08-10.

26. <http://www.kde.org/> – retrieved on 2012-08-10.

3 Challenges

3.5.1 Observation is Not Always Enough

Lansdale (1988) wrote that “underlying psychology of information management cannot be directly inferred from users’ behavior in offices”. The behavior of users doing their workflows on their systems is a result of the tools which may or may not support certain behavior under their current circumstances. When a user study reveals that most users use icon interaction to save the current document, any conclusion that icon interaction is “better” than interaction with keyboard shortcuts or menu entries is not a valid one.²⁷ It still might be the case, but not because of this observation.

In my opinion, some PIM studies made observations and then drew possible unwanted conclusions about what users need in order to achieve the highest level of efficiency or the best user experience. This was based solely on users’ behavior with their current software environment.

Observations can only reveal how users use their current system. This does not necessarily mean that there are not other circumstances, tools, or methods which might lead to better user experience or higher efficiency. Of course, studies based on observations are very important for optimizing current tools! However, the reasoning from participants showing certain behavioral patterns to “this pattern is desirable” is not valid (Lansdale, 1988).

Similarly, comparative studies can only reveal differences of the implementations which were tested. Nothing more, nothing less. Different tools implementing the same methods do not necessarily lead to the same results. Participants are not able to differentiate between implementation and methodical issues. They only use the system as it is.

It is crucial not to draw conclusions which exceed the scope of a study.

²⁷. Lane et al. (2005) covers the topic of the efficiency of icon interaction, menu selection, and keyboard shortcuts.

3.5.2 Long-Term Versus Short-Term

Formal experiments and thinking aloud tests are very important instruments for PIM research. Although, they do have drawbacks. Laboratory experiments with test persons test user behavior in an artificial environment for a very short period of time. Extrapolating results might lead to false assumptions.

If a test person works on a new system in a lab experiment, it is not the general behavior which is observed. It reveals only the first impression and the behavior within *the first minutes* of using the software implementation provided.

This thesis cites a number of long-term studies like: Gibson, Miller, and Long (1998); Boardman and Sasse (2004); Dumais et al. (2003); Furnas, Fake, et al. (2006); Cutrell et al. (2006); Sauermann and Heim (2008); Cho, S. Kim, and Lee (2009); W. P. Jones, Hou, et al. (2010); Whittaker, Matthews, et al. (2011). Such long-term studies have drawbacks as well. Typically, conditions can not be controlled in a strict way. Thus, observations of behavior can not be compared among participants to the same extent as in laboratory experiments.

Nevertheless, field studies are very important to let participants interact with an interface over a longer period of time. Most likely, their behavior changes and adapts over days, weeks, or months. This reveals further observations which address the long-term advantages or disadvantages.

It is easier to conduct a laboratory experiment than a (possibly larger) field study. Both kinds of experiments have their specific qualities. In my opinion, for PIM research, many long-term effects are very important. Thus, researchers should run field studies in addition to laboratory experiments. Comparing short-term experimental results with long-term field study results leads to a much more refined picture of the effects of PIM tools on users.

3 Challenges

3.5.3 Using Only Iterative Processes

As stated in the previous section, many PIM studies observe the current behavior of users. Those results lead to better versions of the current methods or tools. This way, the tools and methods improve with time.

However, as new ideas are developed and new research prototypes are created, they are not as mature as the solutions which have been optimized for a certain period of time. Comparative studies between common, well-established tools and research prototypes using a new paradigm are a double-edged sword. On the one hand, misguiding ideas have to be filtered out in direct comparison. On the other hand, however, those prototypes have a strong drawback.

According to the Monte-Carlo optimization method (K. Binder, 1986), finding the global optimum requires the combination of proposing totally new approaches and successively optimizing this new approach. Only after a certain number of iterations, the new attempt can be proven to be a better solution or not. Figure 3.3 (page 91) and its caption visualizes this issue using a simplified example.

3.6 Research Questions

Based on selected challenges in this chapter, the research questions which are addressed in this thesis are:

3.6.1 How Can Navigational Re-finding Be Improved?

Navigational information re-finding is important and deserves more research focus. The current situation is that most people use the default file system browser of their operating system: Microsoft Windows Explorer, Apple os x Finder, or one of several file browsers for GNU/Linux desktop environments.

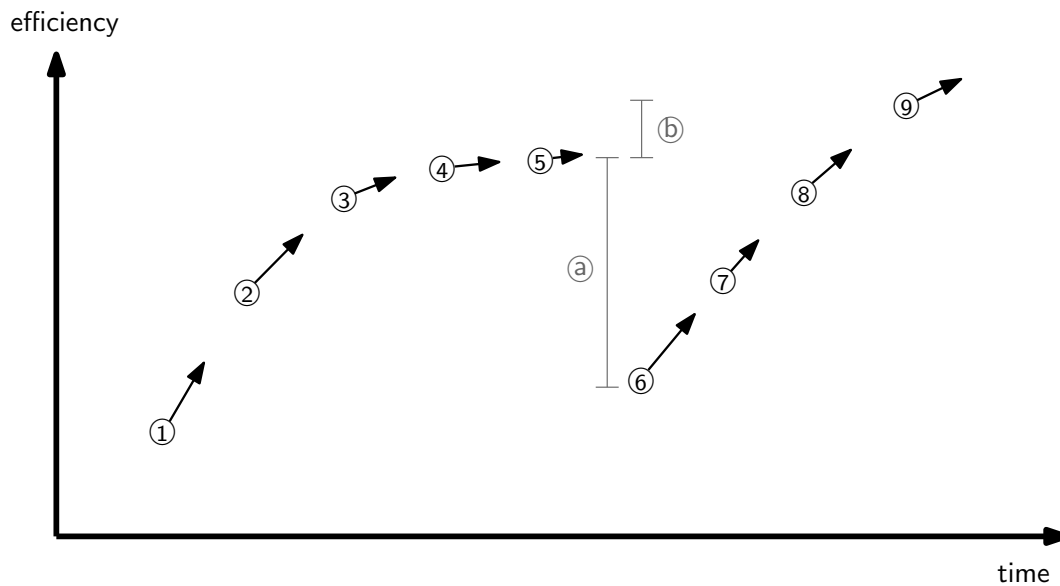


Figure 3.3: A *qualitative* and *simplified* illustration of the tool optimization process. The x-axis symbolizes time, the y-axis is a hypothetical measure of efficiency which is made possible by the tool. ① to ⑨ symbolize state-of-the-art studies which define the efficiency-factor of the tool. The arrows give an impression of how much improvement could be accomplished on the tool which was subject of the study.

① is a tool – the first in its row of studies – which shows room for improvement in a positive direction. An improved version of this tool is evaluated in ②. As the tool is improved much further, studies ③ to ⑤ still show possible optimizations but on a much higher level. The tool seems to be pretty much optimized, big steps of improvement might never happen again. The tool optimization process has become stuck near a *local optimum*.

Then, somebody starts testing a totally fresh approach. Its first study ⑥ reveals a huge drop *a* of efficiency, but it also shows large room for improvements. As this new tool gets improved and studies ⑦ and ⑧ get promising results, the new tool still has a lower efficiency value than ⑤. However, starting with ⑨, the new tool outperforms the old one with *b*. It still might not be the *global optimum* but its local optimum is at least significantly “higher” than ⑤.

The point is that this improvement *b* of the efficiency measure would have never happened if no tool had been proposed which seemed to perform worse compared to the established one. It also needed several iterations and studies to provide as much efficiency as ⑤. With luck, this new tool evolves positively such that it outperforms ⑤ after a certain period of time. Sometimes it is worth trying something completely new. Evaluating only the tools currently in use (like ① to ⑤) may not lead to finding the *global optimum*.

3 Challenges

3.6.2 How to Organize Information In a Better Way?

The classic hierarchy of folders is a limiting factor. A new method using multi-classification, combined with current operating systems and end user environments, could lead to a better PIM experience.

3.6.3 How to Evaluate Such a Method?

Evaluating PIM tools is a sensitive subject. Creating comparative situations for participants is hard. There are also many evaluation methods to choose from. The derived research prototype has to be matched by suitable evaluation methods in order to obtain meaningful results.

I am just a new boy,
Stranger in this town.
Where are all the good times?
Who's gonna show this stranger around?

Young Lust
Pink Floyd

4 The TagTrees Method

The previous chapter listed many PIM challenges. In this chapter, some identified challenges will be addressed by designing a new method for storing and re-finding information. It describes the starting point of this new approach. In accordance with Figure 3.3 (page 91), this new method will be subject to various iterations and optimizations as well. With some luck, this method could be part of many future PIM systems.

4.1 Motivation

Since methods and tools of current computer systems do not support the user in an optimized way, a completely new method might be promising. Implementing and evaluating this new method could result in interesting findings. The task of designing a better method for PIM is hard. Lansdale (1988) described it as follows:

“ But how do we proceed? How do we design for the future?
Ultimately, what we need is to turn this descriptive information
into a specification for future systems.

4 The TagTrees Method

The problem with this is that it is rather like being asked to imagine a colour one has never seen. We have the mental apparatus to see things in terms of how they are, or how they have been, but not how they will be.

It is clear that a single new method can not resemble the best PIM user experience for all requirements and eternity. However, starting from the insights of previous research from Chapter 2, and having in mind the challenges from Chapter 3, each new method does have the chance to be the next step towards a better solution. Each new step brings new insight. And step by step, with each new method developed and evaluated, the PIM user experience becomes further optimized.

So how should a possible next step look like? What do we want to try out next?

From the considerations in the previous chapters, it seems to be the case that following factors need to be addressed in a better way:

- Navigation.
- Multiple [paths](#) to one specific item.
- Multi-classification.
- The role of context.
- Associations.

4.2 TagTrees

The use of [tagging](#) is a promising approach for multi-classification. It is widely adopted in many web-based services, particularly [social tagging](#). Not only social tagging, even a single-user systems can benefit from tagging.

Tagging systems usually use tags for some kind of search-based re-finding interface. If the user enters a tag, the items shown are filtered down to items which were tagged with this selected tag. Navigation structures using tags

are rarely used. Some research tools¹, however, used tags for the purpose of navigation. Inspired by those projects, I derived the following method.

When an item \mathcal{I} is added into the system, the user u assigns a set of tags t_1, \dots, t_n to this item:

$$\mathcal{I} \xleftarrow{u} (t_1, \dots, t_n) \quad (4.1)$$

Using the tags t_1, \dots, t_n , navigational structures are created by the tool. These navigational structures are created to enable the user to navigate to the item using every possible (sub)permutation of its tags. The term “(sub)permutation” here is used in a simplified way. To be precise, this is defined as “the set of the tag-permutations of the number of tags without repetition”.

With n as the number of assigned tags to an item \mathcal{I} , there are $p(n)$ possible paths referring to the item from within the TagTrees:

$$p(n) = \sum_{i=1}^n P(n, i) \quad (4.2)$$

In general, $\mathcal{P}(n, k)$ is defined as the k -permutations of n without repetition²:

$$\mathcal{P}(n, k) = \frac{n!}{(n - k)!} \quad (4.3)$$

This results in the combined formula for the number of $p(n)$ possible paths:

$$p(n) = \sum_{i=1}^n \frac{n!}{(n - i)!} \quad (4.4)$$

The correlation between the number of tags n associated to an item \mathcal{I} and its resulting paths $p(n)$ is exponential as illustrated in Table 4.1.

1. For example TagFS (see Section 2.2.15), hFAD (see Section 2.2.16, or SemFS (see Section 2.2.14).

2. See Kreyszig (1993, p. 1161, Theorem 3).

4 The TagTrees Method

Number of tags n	1	2	3	4	5	6	7	8	...
Number of paths $p(n)$	1	4	15	64	325	1,956	13,699	109,600	...

Table 4.1: Illustration of the growth of the number of paths according to the number of tags used.

For example, if an item was tagged with three tags ($n = 3$), each of the following fifteen paths³ results in a successful retrieval process: $t_1, t_2, t_3, t_1/t_2, t_1/t_3, t_2/t_1, t_2/t_3, t_3/t_1, t_3/t_2, t_1/t_2/t_3, t_1/t_3/t_2, t_2/t_1/t_3, t_2/t_3/t_1, t_3/t_1/t_2$, and $t_3/t_2/t_1$.

Under the assumption that the user does not recall tags which are not associated with the item, she should be able to re-access the item. Each path within the TagTrees can be interpreted as an AND-relation of the related tags forming an implicit filter query. The deeper a user navigates into the TagTrees structure, the narrower this implicit query gets, and the fewer matching items are listed. Within the TagTrees structure there is not a single path that contains no item at all.

Some aspects of this method were influenced by ideas from Bloehdorn et al. (2006) and Mohan, Raghuraman, and Siromoney (2006), although TagTrees does not have the same problems with widowed items or a missing delete method. The TagTrees method was initially published in Voit, Andrews, and Slany (2011).

4.3 TagTrees and File Management

The TagTrees method can be applied to any kind of information management. In order to demonstrate, discuss, implement, and evaluate this method, *we will focus hereafter on its application for filing and re-finding files on a local file system* for a single user.

Figure 3.1 (page 78) visualizes the dilemma of multiple folders matching an item. Using TagTrees this problem can be solved: A user storing an [item](#)

³. Here, the posix path separator is used: “/” (forward-slash).

4.3 TagTrees and File Management

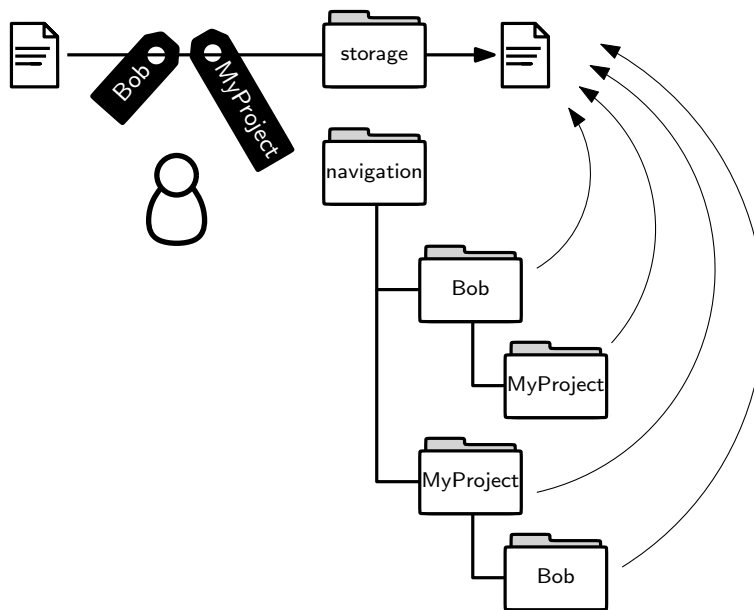


Figure 4.1: The TagTree structure of an example file: After storing a file into the *storage* folder, a user tags it using the two tags “Bob” and “MyProject”. The software then generates the TagTree structure below the *navigation* folder. Each (sub-) permutation of tags leads to a folder which holds a link to the original file in the *storage* folder. The user is able to navigate to this item in many ways. The TagTrees are maintained automatically when the user renames or deletes items in the *storage* folder.

can assign multiple tags to this item. In a dedicated navigational structure, those tags form an automatically generated and updated folder structure. This structure consists of folders resembling every (sub)permutation of tags. In each of those folders, a [link](#) to the original item is created. The name of this link reflects the original item name.

Figure 4.1 shows an example where the root of the TagTree navigational structure is named *navigation*. Two tags were applied to an item stored in *storage* resulting in four additional navigation paths.

To access an item stored with the TagTrees method, a user is able to navigate any combination of tags. This enables [associative navigation](#) depending on the current context of the user.

4 The TagTrees Method

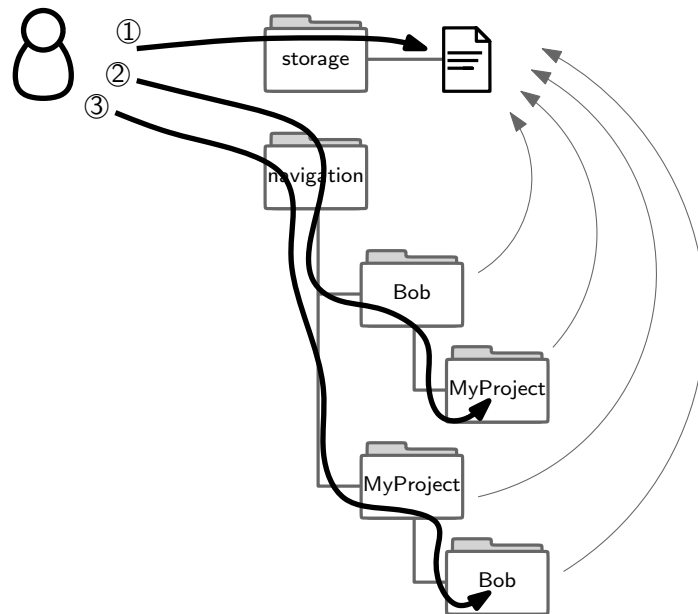


Figure 4.2: Three different examples on how to navigate to an item: Depending on the cognitive state of the user, she is able to navigate many different paths to one single item. She can ① access the item directly in the *storage* folder when she remembers the item name. If her context is based on the person Bob, she might prefer accessing the item using the tag related to Bob ②. In the context of the project MyProject, she probably wants to navigate to the item using its project tag ③.

Figure 4.2 illustrates a small example. The user is able to access the original item directly if she remembers the item name that is stored within the *storage* folder. This is also the recommended location that should be included in any kind of indexing tool such as desktop search engines⁴.

Let us assume that the store in the example belongs to Bob's supervisor, who is preparing for an employee review meeting with him. Most probably she will start her navigation process by opening the tag-folder Bob. In this folder, she recognizes the tag-folder of MyProject. After changing to this folder, she notices the file where Bob summarized some thoughts on his colleagues' project called MyProject. She is able to retrieve his thoughts

⁴. The TagTrees sub-hierarchy should be excluded from backup and indexing.

4.4 TagTrees Addressing Previous Research

and this is a valuable input to the appraisal interview later on.

Another context for the same TagTree structure: the owner of the very same store is not Bob's supervisor. For the next example, the owner of the store is the project manager of MyProject, preparing for a report on this project. This time, she most probably starts by selecting the MyProject tag-folder because this is the focus of her task for now. Within the MyProject folder, the tag-folder Bob gets her attention. By opening this folder she also gets to Bob's ideas about MyProject which leads to useful input for the report.

Those two scenarios show the usefulness of having different navigational paths to the same item. Different context situations lead to different folder selection. As long as the navigational path contains tags from a specific item, the item is found.

The benefit of TagTrees becomes even more clear with one additional notion. Let us assume that Bob is not directly assigned to the MyProject team. Nevertheless, he had some ideas about his colleagues' project and wrote them down. And now reconsider the two scenarios once again. In a strict hierarchy of folders, this file would most probably not have appeared in a navigational process at all. TagTrees support serendipity that would not have occurred with traditional methods.

4.4 TagTrees Addressing Previous Research

In Chapter 2, some papers contained suggestions which might enhance future information management systems. This section refers to some of those suggestions to underline the grounding of the TagTrees method in previous research.

Cutrell et al. (2006, pp. 9, 10) mentioned critical "influence points": "for files it is during the 'Save dialog' when users are thinking about how the document will be recovered later." This emphasizes the importance of implementing the tagging process correctly when a new item is filed.

4 The TagTrees Method

Among other conclusions, Malone (1983) mentioned three things to “[make] the mechanical process of creating multileveled classification systems very easy”:

1. Multiple classification.
2. Deferred classification.
3. Automatic classification.

Multiple classification⁵ is made possible by applying tags instead of storage paths within a strict hierarchy. The possibility to defer the classification of an item could be solved by not forcing the user to tag an item at any certain moment. To support automatic classification, an implementation can make use of implicit information such as date of storage, item size, and item type. Additionally, a tag recommendation system is able to support users in the process of applying tags.

Lansdale (1988) contains several notions which I addressed by TagTrees. For example, the following quote points out the importance of context:

“ [A]t the time of recall, the context of what we are thinking about provides a prompt for memory. If this context matches the way in which the information was initially interpreted, then recall is successful, but if it does not, then recall fails.
[...]
[M]ost of what people remember about a particular document is partial: they remember some of what was associated, but not all. Systems which require complete accuracy (memory for all three elements) therefore fail.

For re-accessing an item within TagTrees, a user does not have to remember the order of the tags, or the complete set of tags.

However, associative navigation is not only mentioned in Lansdale (1988). Many other papers address this need. J. Kim (2011) evaluated an associative browsing model (see Section 2.2.19) and showed that “associative browsing provides an effective and efficient alternative when keyword search fails.”

5. Multiple classification is an important issue in Malone (1983); Lansdale (1988); Nielsen (1996); Quan et al. (2003); Shirky (2005); Weinberger (2007); Perugini (2009) and many more.

4.5 The Effort of the Storage Process

In Gifford et al. (1991) the authors produced a semantic file system which provided associative navigation. Collins (2011) developed an interface for tabletops, which implemented associative navigation among other things. Elsweiler (2007) emphasized the importance of these issues for PIM as well. In Chau, Myers, and Faulring (2008b) the authors developed and evaluated the Feldspar tool (see Section 2.2.17), which also allows associative navigation. Also, Bush (1945) mentioned “a mesh of associative trails” in his famous article “As We May Think”. Trattner et al. (2012) found that users tended to use several different paths to navigate from one source item to a destination item.

The test persons in a study by Boardman and Sasse (2004) cited three main reasons for failing in information retrieval:

1. Removing items from the set of accessible items (deleting or archiving).
2. Clutter.
3. Misfiling.

The additional re-finding capabilities of TagTrees might reduce the problem of having too many items within a folder. By selecting specific tags for items to access, the average result set – showing only matching items – might lead to less clutter in the retrieval process.

When a test person mentions misfiling as a reason for failure to access an item, she most probably refers to the fact that cognitive models change (as described in the paragraph on page 79 of Section 3.4.1). The test person presumably had good reasons to file the item into its storage folder. While in the context of filing this item, the selected storage folder made sense. Otherwise the user would have chosen another folder. The multi-classification of TagTrees is able to improve the situation here as well.

4.5 The Effort of the Storage Process

This section discusses the implications of the TagTrees method and compares it to the traditional method of [folder hierarchies](#). For the sake of

4 The TagTrees Method

simplicity, this section mostly refers to *files* as items, neglecting the fact that folders could also be managed within TagTrees.

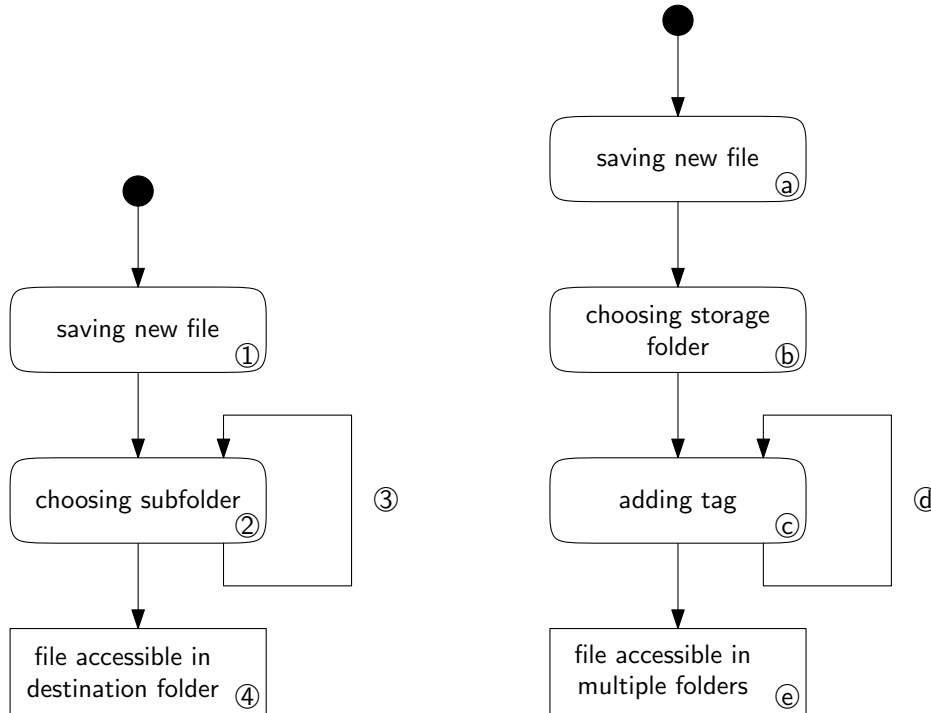
Figure 4.3(a) shows a simplified flowchart of the process when a user stores a file on her local hard disk. While in an application, where the file was created or loaded from external sources, the storage process is initiated. Then a “Save file...” dialog window will be presented to the user. Within this dialog window, the user navigates through her hierarchy of folders looking for a promising destination folder. Due to the fact that the number of files we handle is constantly increasing (see Section 3.1.1), there is a certain probability that the destination folder moves deeper and deeper into our folder hierarchy over time. Having decided where to locate the new file, the filing process is finished resulting in a single folder to look for the file.

To integrate TagTrees into our current computer systems without modifying the “Save file...” dialog windows of our current applications, there has to be at least one additional step, as shown in Figure 4.3(b). When a user wants to store a new file using TagTrees, there has to be a storage folder for physically storing the new file. This storage folder has to be monitored by a TagTrees implementation. Upon detecting a new file, a dialog window would appear, where tags can be associated to the new file. After all tags are entered and confirmed by closing the tagging dialog window, the navigational structures of the TagTrees will be generated. This process results in the file being accessible via a large number of paths using associative navigation.

It is easy to see that the TagTrees method does not simplify the filing process. Instead, the tagging step has to be done in addition to choosing a destination folder. The clear benefit of TagTrees lies in the re-finding process which is enhanced by multiple paths.

However, a clever implementation of TagTrees and a suitable setup are able to compensate for the additional storage effort. First of all, the storage folder of a TagTrees store should be chosen so that it is easy to reach in any save file dialog. This way, there is no need to navigate to a folder which is deep down in the folder hierarchy. In this case, choosing the storage folder for a TagTree store most likely means less effort than searching for

4.5 The Effort of the Storage Process



(a) A simplified flowchart for adding a file to the local folder hierarchy. In ② and ③, the user has to navigate through her local folder hierarchy, searching for a suitable destination folder for the file. The process results in ④, where the file is accessible in the single destination folder.

(b) A simplified flowchart for adding a file to a local TagTree storage. It is assumed that in b, the storage folder is easy to reach (Desktop, Home, ...) and does not require a long navigational process. The steps c and d require one or more tags to be entered. Several features offer good usability and convenience: [tag completion](#), [tag recommender system](#), and default tags. The resulting TagTrees in e provide multiple representations of the new file.

Figure 4.3: Simplified flowcharts of the process of adding a file to the local file system. The steps ① and a can result from a download process, a copy process, or from generating a completely new file in any user application. In comparison, the tagging process contains one additional step. This might be compensated for by the enhanced re-finding possibilities and the fact that files are usually stored only once, but accessed more than once.

4 The TagTrees Method

a destination folder in the traditional folder hierarchy. There is actually an advantage for the TagTrees method in this step of the process.

The following step of tagging the newly stored file is a crucial one in terms of implementation issues. Many tagging systems that were subject of PIM studies lacked even basic features for a convenient tagging experience. Things like automatic completion of already known tags and a tag recommendation system can ease the tagging process dramatically. Usability has to be optimized in order to make this step as easy and rewarding as possible.

Tagging an item has several benefits compared to choosing a destination folder deep down a hierarchy of folders. In a tagging dialog, users are able to assign any possible tag directly. This is not possible in a filing situation where a user might be forced to navigate her up and down the hierarchy when there is no obvious destination folder.

Additionally, a user does not have the possibility to use a folder recommendation system for traditional filing in a folder hierarchy. This process hardly scales over time. Most used tags and other recommended tags can be easily selected using the mouse and do not require typing or navigation.

Any implementation of the TagTrees method has to make sure that the tagging dialog receives great attention in the design process. Tagging is often seen as a burden. If tagging is done correctly, the many advantages while accessing items could be noticed by the user.

First of all, TagTrees support serendipity such that every now and then items appear which were already forgotten. Because of the many navigational paths, this most likely happens more often within TagTrees than in a strict hierarchy of folders. If files are stored in the wrong folder, they are seldom found anymore by using navigation. As Section 3.1.1 described, most files are never accessed at all. There is a chance that associative navigation can lower the percentage of lost files.

The studies by Gibson, Miller, and Long (1998), Leung et al. (2008), and Cho, S. Kim, and Lee (2009) clearly showed that the number of files accessed is a multiple of the number of files newly created or stored. Accessing files is a task which definitely happens more often than storing new

4.5 The Effort of the Storage Process

files. So if the item access process is improved by using associative navigation in TagTrees, the higher effort of storing still results in an overall improvement for the user.

The fact that files are stored ever deeper in the folder hierarchy (Cho, S. Kim, and Lee, 2009, p. 163) requires more effort, because users take longer to navigate to these files. From this point of view, a relatively flat TagTrees structure is an advantage. The user just has to navigate to as many tag-folders within the TagTrees as needed to locate the desired item among the items displayed. Starting from visiting the very first tag-folder, she has the chance to locate the item because items from “deeper” levels also appear in the “higher” levels of TagTrees.

Another important aspect, which might lead to a higher acceptance for tagging, is also related to the long-term benefits. Anecdotal evidence of long-term users of TagTrees shows that users start to appreciate this method much more by having positive moments in a re-finding process. However, this requires a change of mind. Users are so accustomed to the traditional folder hierarchy method, that they tend to use TagTrees the same way in the beginning. When they learn that there is no need to remember the tags they chose during the storage process – like they need to remember the exact storage path in the traditional method – opinions and behavior change. Users start having confidence that any association, which comes up when thinking of an item will bring them to this item. In my opinion, the use of a well-chosen [controlled vocabulary \(cv\)](#) amplifies this effect even more.

His dreams are like commercials
But her dreams are picture perfect and
Our dreams are so related
Though they're often underestimated

Bubble Toes
Jack Johnson

5 tagstore Implementation

To be able to evaluate the assumptions of the previous chapter and to compare with previous research, the TagTrees method was implemented. The research software which was developed is called *tagstore*.¹

This chapter describes the general internal structure of the implementation, describes every module in detail, and illustrates the different setup possibilities. The internal data structure consisting of human-readable text files, is explained. Furthermore, the most important user-related workflows are demonstrated, best practices are listed, and technical limitations of the current implementation are discussed. The final section of this chapter contains links to similar tagging products.

The TagTrees method and the tagstore implementation were also described in Voit, Andrews, and Slany (2011), Voit, Andrews, Wintersteller, et al. (2011), and Voit, Andrews, and Slany (2012a).

1. *TagTrees* (the method) is always written with two capital T's and *tagstore* (the implementation) is always written in lower case letters.

5.1 Motivation

The TagTrees method is a concept which is of limited use, if it can not be used for testing and evaluation. In order to obtain both subjective (opinions, suggestions,...) and objective (performance numbers, tagging behavior, navigation behavior,...) feedback, the TagTrees method was implemented in tagstore.

As an additional benefit, people using tagstore are acquainted with a different viewpoint. In contrast to the more or less static hierarchy of folders, dynamic TagTrees allow associative navigation. I think that this minor notion has huge potential, because users of tagstore are made aware that there *are* possible alternative ways of file management.

Prior to the start of development of the current version of tagstore, some basic attributes were specified:

Minimal task effort Each additional step a user has to do has to be well justified. Workflows or features requiring too many steps without direct benefit to the user will not be used. Therefore, the tagging process has to be designed in a way that the user perceives minimal task effort and being supported by the software as much as possible. The highest possible level of usability is a key issue here as well.

Compatibility To be able to do field studies and give away tagstore to anybody who is interested, tagstore should not be an “island”. Users should be able to use any software they were using before tagstore. In particular, they should be able to continue to use the file browser, desktop search engine, and backup software of their choice. Another level of compatibility is platform independence. If possible, the use of tagstore should not be limited to one operating system alone. That said, the tagstore implementation should be kept independent of platform-specific issues.

5.1 Motivation

Transparency Users do not trust in software products which confuse them. The TagTrees method alone should be the only new thing around. Fiddling with database systems², hiding the metadata in some proprietary format, using too much “hocus-pocus” of any kind is not desirable. Configuration data and metadata is stored in human-readable text file format. Users of tagstore should always know where their data is and that nothing unexpected will happen to it. In case of an emergency, they should still be able to use it without any part of tagstore.

Openness The source code of tagstore is under an open-source license³ and available for free download^{4, 5}. Additionally, all research data that was generated in studies and experiments is available for download as well. This ensures that everybody is able to verify, re-create, or extend my research in the spirit of Open Science⁶ and *Reproducible Research*⁷.

Limitations To follow the principle of *keep it short and simple (KISS)*, tagstore does not have to be *the* perfect product. Usually, research prototypes have to be only good enough for testing. The tagstore implementation should exceed this level, without trying to become a flawless product. With these restrictions in mind, it is possible to implement tagstore even with the given, very limited resources. The self-defined area of interest of *single user, single system* helps to prevent losing track with complicated security or multi-user issues.⁸

2. Most likely, databases break at least the usual backup method (shut down database, do backup, restart database) and cause users to duplicate data just in case they are not able to access this database black box for any reason.

3. The license for tagstore is [GNU General Public License \(GPL\)](https://www.gnu.org/copyleft/gpl.html) version 3 or later: <https://www.gnu.org/copyleft/gpl.html> – retrieved on 2012-08-16.

4. <http://tagstore.org> – retrieved on 2012-07-17.

5. <https://github.com/novoid/tagstore> – retrieved on 2012-08-16.

6. <http://www.openscience.org> – retrieved on 2012-08-16.

7. <http://reproducibleresearch.net> – retrieved on 2012-08-16.

8. Further research has to be done to explore this very important issue of multi-user aspects of TagTrees.

5 tagstore Implementation

Framework One of the most important reasons for implementing this software is for testing purposes. Therefore, tagstore has to provide features that enable test persons to modify interface behavior and configuration without having to change a single line of source code. To be able to test different things, tagstore has to implement more than the standard TagTrees method described in the previous chapter. This way, tagstore can be seen as a testing platform, a framework for doing a wide spectrum of tests in similar research areas.

5.2 Software Modules

In order to stay at a high level of implementation⁹, tagstore was developed with Python¹⁰ version 2.7. For the **Graphical User Interface (GUI)** elements, the Qt framework¹¹ was used. As a very positive side effect, these changes also provide a certain level of independence from the underlying operating system.

As illustrated in Figure 5.1, tagstore consists of four levels of abstraction. The following sections describe each of the modules of tagstore.

tagstore Dialog and tagstore Manager These two modules are the only modules that have a GUI. Most of the functionality of tagstore is accessed through these modules. Section 5.3 and 5.4 describe those modules in detail.

tagstore Logic The next module of tagstore (as shown in Figure 5.1) is *tagstore logic*. This module and all lower-level modules do not directly have a GUI. Following the principle of **Model-View-Controller (mvc)**, the central logic behind tagstore is encapsulated in this module: controlling the GUI, managing the data.

⁹. The proof of concept implementation was finished within only two days.

¹⁰. <http://www.python.org/> – retrieved on 2012-08-16.

¹¹. <http://qt-project.org/> – retrieved on 2012-08-16.

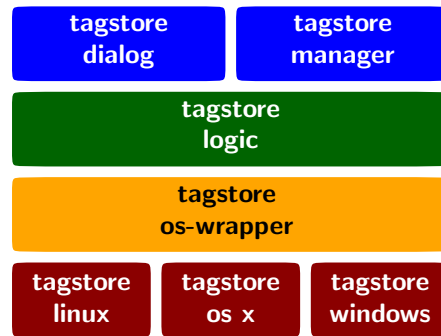


Figure 5.1: The tagstore implementation layers: the user interface consists of the *tagstore dialog* window for tagging items and the *tagstore manager* for maintaining the stores. In the *tagstore logic* layer, the general algorithms are implemented. General adaptations to the operating systems are kept in the *tagstore os-wrapper* layer. Operating system specific implementation is encapsulated within *tagstore linux*, *tagstore os x*, and *tagstore windows* packages.

tagstore OS-Wrapper The os-wrapper module provides generic interfaces to the modules that are specific for the target operating systems: GNU/Linux, Apple Mac os x, and Microsoft Windows. All system calls are abstracted to this level of access.

tagstore Linux This module contains all system calls that are specific to GNU/Linux. TagTrees are created using folders for each tag and [symbolic links](#) to link the items in the TagTrees with their originals in the storage folder.

tagstore OS X This module contains all system calls that are specific to Apple Mac os x. The implementation is very similar to the module for GNU/Linux, and symbolic links are used here as well.

tagstore Windows This module contains all system calls that are specific to Microsoft Windows. Unlike the other two operating system-dependent

5 tagstore Implementation



Figure 5.2: A screenshot of the tagstore tagging dialog from the 2012-08-16 version using GNU/Linux. On the right-hand side, the list shows all untagged items that are stored in the storage folder. At the bottom right, the user can enter comma-separated tags into the tag line. The tag-recommending feature shows the two possible completions for partially typed-in string “hi”. The left-hand side consists of tag recommendations from the recommender system.

modules above, there are no usable¹² links in NTFS. Instead, shortcuts are used. Owing to the technical representation of shortcuts in the user space and not at the file system level, their performance is very poor compared to symbolic links. See Section 5.8.2 for more information concerning performance.

5.3 tagstore Dialog

The most prominent module, as seen from the users’ perspective, is the dialog window that offers the tagging capability. Figure 5.2 shows an example tagstore dialog window with one item to tag and the user already typing the second tag. The following elements are visible:

12. Microsoft Windows uses New Technology File System (NTFS) as its file system. Although NTFS offers technical representations of links, Windows does not offer any usable user-space tools to maintain links. Instead, Windows uses so-called shortcut files. All other operating systems use symbolic links implemented at the file system level.

Item list On the right-hand side there is a list containing all untagged items. When a user chooses to press the “Postpone” button, current items remain in the list for future tagging.

Tag line The tagging line (or short: “tag line”) is below the item list. Users can enter tags for the currently selected item in the item list. Multiple tags are separated with commas (“,”). Thus, multi-word tags are possible with tagstore as well. There can also be a second tag line for a different type of tags. See page 126 for details on this.

Tag cloud The left-hand side is filled with a tag cloud. This cloud shows a subset of already known tags. The bigger the tag is visualized, the more relevant the tag is according to the recommendation system. Users are able to click tags in the tag cloud, which are then appended to the tag line. See Section 5.3.1 below for more details on the recommender system.

Buttons The “Tag!” button assigns the tags from the tag line to the currently selected item on the item list. If this was the last item within the item list, the tag dialog is closed. Selecting the “Postpone” button closes the tag dialog immediately. In this case, all items which are listed in the item list remain there. The “Manager...” button opens the tagstore Manager. To its left side, there is the “Help” button, represented by a question mark icon. In any tagstore interface, the help button opens a help window which explains all relevant GUI elements and their functions.

Postponing the tagging process is a widely demanded notion that most PIM systems do not provide yet. Malone (1983, pp. 8, 11) mentioned “untitled piles” and “deferred classification”. Whittaker and Sidner (1996) described the problems of users who want to “postpone their judgement in order to determine the value of information”.

Another feature of tagstore is very convenient for tagging several items: When the user has filled the tag line and confirms with “Tag!”, the current list of tags stays in the tag line (*default tags*). If the next item is similar to the previous one, re-using tags is made very easy.

5 tagstore Implementation

Tagging multiple items with the same set of tags is intuitive: using the usual method to mark items¹³ defines the set of items. When the user then adds her tags to the tag line, the set of items is tagged using the same tags in a single step.

The tagging dialog as well as the interfaces of the tagstore Manager were initially designed using the paper prototype method. Over time, they were constantly optimized using the feedback of long-term users and observations from formal experiments. The usability of the tagging dialog is crucial for the user acceptance.

5.3.1 Tag Recommendations

The tag recommendation system (or in short “recommender”) uses several aspects to generate the tags of the tag cloud:

Usage Frequency The more often a tag is used, the higher its ranking becomes. More frequently used tags are likely to be more important to the user.

Similarities to Item Name The name of the item is split into sub-words. Each sub-word exceeding a length of four is compared to the set of known tags using the Damerau-Levenshtein distance. If a sub-word of the item name is equal to a tag, its value is maximized.

Type Similarity The algorithm determines tag usage frequencies related to [file extensions](#). For example, image files with the [file extension](#) “.jpg” are likely to receive tags similar to other image files of that kind.

¹³. Selecting single items from the item list: Ctrl and the primary mouse button; selecting a range of items from the list: starting with the currently selected item using Shift and the primary mouse button.

Similarities to Known Items The sub-words mentioned above are compared to all sub-words of known items. That way, similarly-named items receive similar tag recommendations.

Metadata The metadata of an item (content, explicit metadata, ...) should be part of any advanced recommendation system. Unfortunately we faced a problem: when adding huge files like movies to tagstore, metadata could only be extracted when the file storage process is completed. On the one hand, this is not trivial to detect.¹⁴ On the other hand, the tagstore dialog would not appear instantly when the file is initially added, but when the file copy process has finished. This would result in interrupting tagstore dialogs when the user is already in a different context. Therefore, we skipped file content for this implementation.

The recommender modules result in a set of values for each tag. After generating the sum of all values per tag, the highest ranked tags are displayed in the [tag cloud](#).

The implementation of the recommender is described in detail in Schober (2012).

5.4 tagstore Manager

The second user interface of tagstore is called the *tagstore Manager*. This GUI is to configure general tagstore settings and stores. It is important to mention the fact that a user can have as many stores as needed. For example, one store for personal items, one for each business project, and one store for movies. There is no connection between stores besides the fact that they are all managed by the same tagstore installation. Each store has its own storage folder and its own TagTrees.

The tagstore Manager consists of several different views which are shown as tabs. Each tab comprises a configuration dialog of a feature of tagstore.

¹⁴. Determining the status of a file transfer over the network is hard to do, because transfer can be delayed due to network bandwidth and many other issues.

5 tagstore Implementation

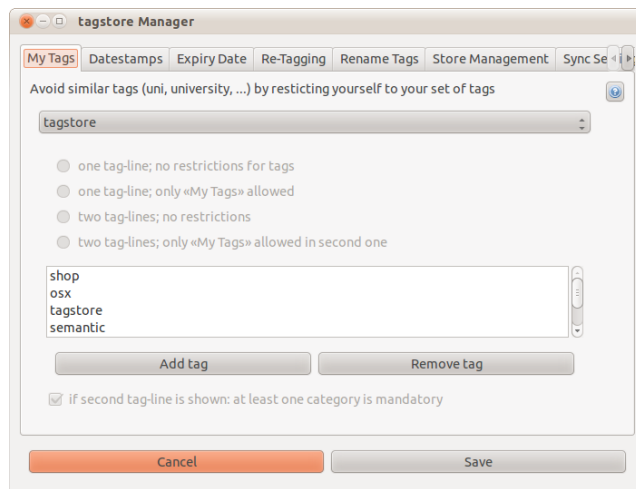


Figure 5.3: tagstore Manager – My Tags.

Some of them are for user comfort, and some of them are for basic store management features.

5.4.1 My Tags

The *My Tags* tab (Figure 5.3) configures a [controlled vocabulary](#). This *store-specific*¹⁵ feature can only be activated before a store is created. For all stores that have this feature activated, the list of tags can be managed here using “Add tag”, and “Remove tag”.

5.4.2 Datestamps

Temporal views are very important to users, as many studies described in Chapter 2 show. In tagstore, the user is able to add tags which are datestamps. This store-specific optional feature can be very handy: an activated

¹⁵. All features which can be used on a per-store basis offer a store selection drop-down list at the top of the tab window.

5.4 tagstore Manager

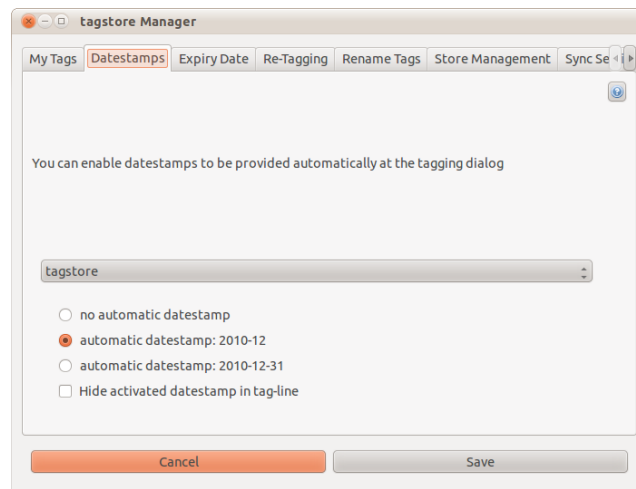


Figure 5.4: tagstore Manager – Datestamps.

datestamp is a default tag which appears in the tag line (Figure 5.4). Either the current day (for example: 2012-08-16) or the current month (for example: 2012-08) is used in ISO 8601¹⁶ format. These default tags can be manually modified or removed by the user. Each item tagged with such a datestamp tag, can be found in corresponding TagTrees folders. Datestamp tags are tags just like any other tag.

User comments showed that using datestamp tags for months is a reasonable choice: it does not lead to as many tags as the daily datestamps do. Within the TagTrees of 2012-08, the user finds all items (and tags of items) which were added in August 2012.

The “Hide activated datestamp in tag-line” option does not place the datestamp into the tag line. It just uses this tag silently, without distracting the user with its presence in the tag line.

5 tagstore Implementation

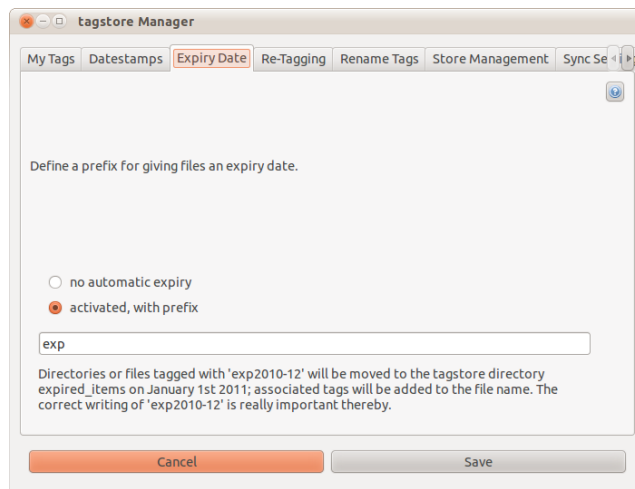


Figure 5.5: tagstore Manager – Expiry Date.

5.4.3 Expiry Date

In my opinion, many more applications should provide a feature comparable to the *Expiry Date* feature (Voit, Andrews, and Slany, 2009). As a general tagstore property, the user is able to activate and configure this feature in this tagstore Manager tab globally (Figure 5.5). The string entered into the input line works as a prefix. Combinations of this prefix and a valid ISO 8601 datestamp for months are interpreted as expiry dates (by convention).

For example, if the user chooses to stick with the default expiry date prefix exp, a tag like exp2013-01 is an expiry date. This tag is entered just like any other tag: in the tag line of the tagstore dialog. Hence, the user is able to navigate to items related to a certain expiry date in the TagTrees.

Items, whose expiry dates are in the past, are moved to the storage folder named expired_items. All tags of those items are added to the end of the file name. Thus, items are never deleted, just moved to this special folder. The user is still able to recover the items in the expired_items folder

16. http://www.iso.org/iso/catalogue_detail?csnumber=40874 – retrieved on 2012-08-16.

5.4 tagstore Manager

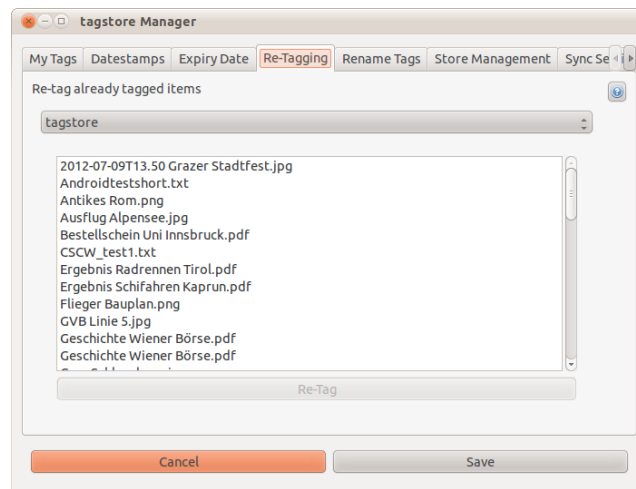


Figure 5.6: tagstore Manager – Re-Tagging.

or delete them manually. Using expiry dates on items keeps the TagTrees leaner.

Almost no user regularly walks through her folder hierarchy to look for items that are of no use anymore. The big advantage of this feature is that the user has a very good feeling for any “expiration” (only) *during the process of storing*. If the user is planning a trip to Paris, she can tag the [portable document format \(PDF\)](#) file of the Paris Métro¹⁷ with an expiration date that ends after her trip. This way, this item is removed from her direct view (TagTrees) automatically. She does not have to remember to delete unnecessary items any more. If the user knows that an item will expire, but is unsure what expiry date to choose, she can choose an expiry date in the far future. This is still better than collecting items that are of no value any more.

5.4.4 Re-Tagging

If the user wants to modify the tags associated with an item, she can go to this tab of the tagstore Manager (Figure 5.6). For each store, there is an alphabetically ordered list of all items. After selecting an item and clicking on “Re-Tag”, the tagstore dialog window opens with the corresponding item and its tags. As usual, the user is able to modify tags in the tag line and confirms with “Tag!”.

From the usability perspective, it would be much better if the user were able to invoke the re-tag process while seeing the item in her file browser. Unfortunately, such a feature is platform-dependent and file-browser dependent. Nevertheless, it can be implemented any time: tagstore comes with the script `tagstore_retag.py` which can be used to invoke such a re-tag process from external tools. If a file browser is able to start external tools, such integration is easily configured.

5.4.5 Rename Tags

Once in a while a user wants to re-organize her tags. This process is called *tag gardening* (Peters and Weller, 2008), and involves deleting, splitting, combining, and renaming tags. For renaming tags, the tagstore Manager offers a separate tab (Figure 5.7). When a user selects a tag and “Rename”, she can change the name to a different one. This way, tags can be both combined (renaming to an existing one) or renamed. All items associated with this tag are re-assigned automatically.

For example, tag gardening is recommended in the following cases:

- Reclining interest in a topic: combine several old tags into one:
 - oldtimers, vans, convertibles → automotive
- Growing interest: split tags:
 - IT → hardware, software, internet
- Improving usability for typing frequently used tags:

17. <http://www.parismetro.com/> – retrieved on 2012-08-16.

5.4 tagstore Manager

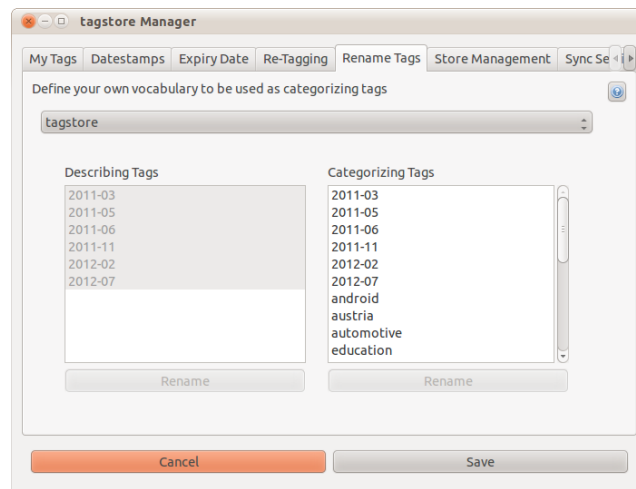


Figure 5.7: tagstore Manager – Rename Tags.

– university → edu

5.4.6 Store Management

The store management tab of the tagstore Manager (Figure 5.8) is needed when a new store should be created or an old one deleted. As mentioned before, a user is able to maintain as many different (and separated) stores as needed. Best practices show that separation into different stores is only reasonable, when two topics have little or nothing in common and the cognitive model differs as well.

When a new store is created (“New Tagstore” button), its configuration is taken from the file `store.cfg.template`¹⁸. This is on purpose: test users are not able to change the setup chosen by the persons conducting an experiment. If a person wants to change these settings, any text editor can modify its content.

¹⁸. This template file is located in the installation folder of tagstore: `tagstore/tsresources/conf`.

5 tagstore Implementation

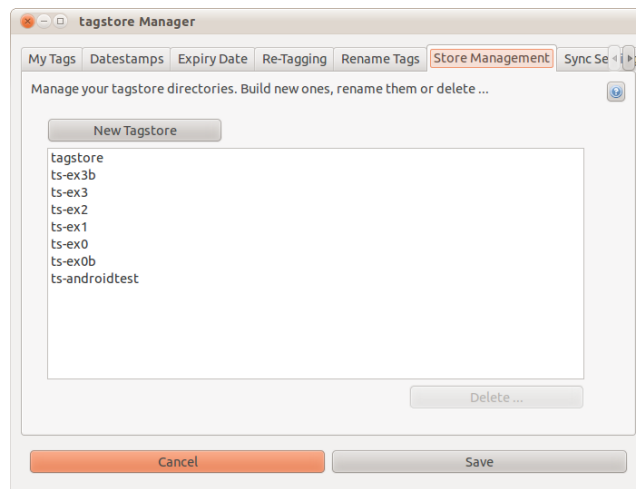


Figure 5.8: tagstore Manager – Store Management.

If an old store is deleted (selecting store and “Delete...” button), all items remain on the hard disk. No item can be deleted by accident. Once again: tagstore never deletes any item from the file system.

5.4.7 Sync Settings

Figure 5.9 shows the tab for setting up synchronization. This feature is currently in beta status. Johannes Anderwald developed a version of tagstore for mobile devices running the Android operating system (Anderwald, 2012). In this tab, the user is able to choose a sync tag. This sync tag is used to determine which items to synchronize to the mobile device. With this method, the user has the ability to choose the items for her mobile device.

For example, with the sync tag android, each item that will be tagged with android gets synchronized to the mobile device. Besides this synchronization, the sync tag is a tag like any other tag.

5.5 Internal Storage Structure

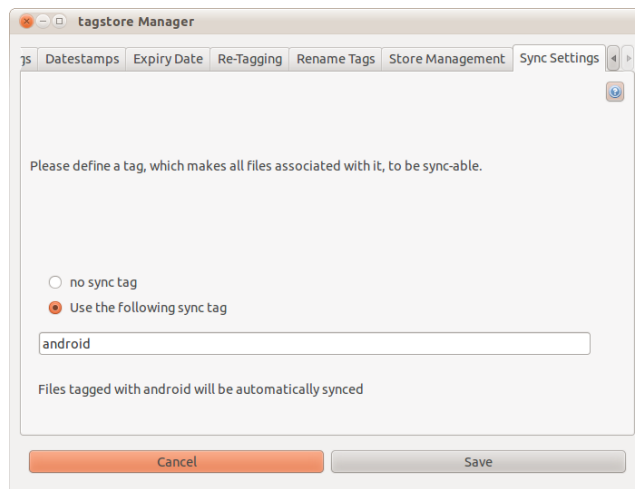


Figure 5.9: tagstore Manager – Sync Settings.

5.5 Internal Storage Structure

The tagstore implementation stores information in a set of text files. Some of them are general settings and stored in a sub-folder of the tagstore installation folder (which is denoted as `$mybin` for the purpose of this description). Other settings are specific to a store, located in a sub-folder of a store folder (denoted as `$mystore`).

Unless stated otherwise, all files are human-readable text files following the standardized format of INI files¹⁹. They can be viewed and modified with any text editor and parsed easily using existing programming libraries.

`$mybin/tagstore/tsresources/conf/tagstore.cfg`

This is the main configuration file for the tagstore installation. An example `tagstore.cfg` file content might look like this:

19. https://en.wikipedia.org/wiki/INI_file – retrieved on 2012-08-17.

5 tagstore Implementation

```
[settings]
config_format=1
current_language=en
store_config_directory=.tagstore
store_tags_filename=store.tgs
store_config_filename=store.cfg
store_vocabulary_filename=vocabulary.txt
tag_separator=","
supported_languages="en,de"
expiry_prefix=exp
max_tags=7
sync_tag=android
first_start=false
show_wizard=false
show_tag_help=false
show_my_tags_help=false

[stores]
1=/home/user/private
2=/home/user/job/aproject
3=/home/user/job/otherproject
```

`config_format` is a version number of the format of the configuration file content style. This number will be changed if the format of this file is changed substantially.

`current_language` defines the language of the tagstore implementation. Currently, tagstore is implemented multilingually in English (value: `en`) and German (value: `de`). The setting is derived from the system settings, the default and fall-back is English.

`store_config_directory` defines the name of the folder in the store which contains internal data.

`store_tags_filename` defines the name of the file which holds the item names, assigned tags, and a timestamp.

`store_config_filename` defines the name of the file which holds the store settings.

5.5 Internal Storage Structure

`store_vocabulary_filename` defines the name of the file which holds the [controlled vocabulary](#) (“My Tags”) of a store (see [Section 5.4.1](#)).

`tag_separator` defines the character that separates multiple tags.

`expiry_prefix` defines the value for the prefix of expiry dates. See [Section 5.4.3](#) for more information related to expiry dates.

`max_tags` sets the limit for the maximum number of tags. See [Section 5.8.3](#) for details.

`sync_tag` holds the value for the sync tag, as described in [Section 5.4.7](#).

`first_start`, `show_wizard`, `show_tag_help`, and `show_my_tags_help` hold Boolean values which define whether help and/or wizard windows should be displayed or not.

The list of entries in the `[stores]` section is the list of active stores.

[\\$mybin/tagstore/tsresources/conf/store.cfg.template](#)

This file is the configuration template for the next store which is created. See the description of its implications in [Section 5.4.6](#). The content of this file corresponds to `store.cfg`, which is described below.

[\\$mybin/tagstore/tsresources/conf/store.tgs.template](#)

This file is used as a template for `store.tgs` and contains more or less its empty header.

[\\$mystore/.tagstore/store.cfg](#)

Any configuration which relates to a specific store is stored in this file. A typical `store.cfg` might look like the following:

5 tagstore Implementation

```
[store]
store_id=0

[settings]
config_format = 1
datestamp_format=1
show_category_line=3
category_mandatory=false
datestamp_hidden=false
```

The `store_id` holds the store's identifier number. It corresponds to a line in the `[stores]` section within `tagstore.cfg`.

As already described above, the `config_format` is a marker for the format of this file.

`datestamp_format` can be either 1 for the current month (like: 2012-08) or 2 for the current day (like: 2012-08-17). See Section 5.4.3 for details.

`show_category_line` defines the number of tag lines used and whether "My Tags" is activated:

<code>show_category_line</code>	tag lines	"My tags"
0	one	not activated
1	two	not activated
2	two	activated
3	one	activated

`category_mandatory` is only used for configurations with two tag lines. If this value is set to true, the second tag line is mandatory as well. If it is set to false, the second tag line could be left empty.

`datestamp_hidden` holds the setting for showing the optional datestamp in the tag line. See page 5.4.2 for more details.

`$mystore/.tagstore/store.tgs`

The `store.tgs` contains information which describes the current state of the store content: all items with their tags and the time stamp of their first occurrence. An example file for three items could look like the following:

```

[settings]
config_format=1

[files]
test.txt\tags="test,2011-05"
test.txt\timestamp=2011-05-25 16:38:21
test.txt\category=2011-05
inbox.org\tags=2011-06
inbox.org\timestamp=2011-06-15 08:26:04
inbox.org\category="2011-06,gtd,emacs"
references.org\tags=2011-06
references.org\timestamp=2011-06-15 08:26:04
references.org\category="2011-06,gtd,emacs"

```

[\\$mystore/.tagstore/store.log](#)

All logging information of a store is written in this file. It basically resembles notes about the history of a store: when a new item is added, which tags were assigned, and any incident reported by the tagstore software. This file is of interest for debugging purposes.

This file is not in INI file format.

[\\$mystore/.tagstore/vocabulary.txt](#)

All phrases of the [controlled vocabulary \(cv\)](#) ("My tags") are stored in this file. It is not in INI file format: each line consists of one tag for the cv. The simple format of this file ensures, it is easy to generate from existing cvs and convenient to re-use.

5.6 Workflows

The implementation of TagTrees can be carried out in many different ways. This section describes the workflows from the users' perspective. A number

5 tagstore Implementation

of smaller workflows was already mentioned in Section 5.3 and Section 5.4 and is omitted here.

5.6.1 Installation

The installation files of tagstore can be obtained from the tagstore website²⁰ or from the code repository²¹. There are two different kinds of installation processes: (1) using installer packages which are platform-specific and (2) manually installing tagstore from the source files. The latter is only recommended for users with advanced knowledge²² of their system.

Installer Packages

The installer packages for Microsoft Windows and Apple os x were created using BitRock InstallBuilder²³. All dependencies are included, but can be left out in the installation procedure. The install process is very easy and is accomplished with dialog windows describing the current step and offering help.

Manual Installation

Using the manual installation process ensures using the latest source code available. Obtaining the source code from the repository is explained on github²⁴.

Before installing tagstore, the user has to install the dependencies which are required to run tagstore:

20. <http://tagstore.org> – retrieved on 2012-08-18.

21. <https://github.com/novoid/tagstore> – retrieved on 2012-08-18.

22. Editing configuration files, setting execution paths, installing dependencies, debugging configuration problems.

23. <http://installbuilder.bitrock.com/> – retrieved on 2012-08-18.

24. Basically it can be done by downloading a ZIP archive or cloning the repository using the git version control system.

- Python²⁵ version 2.6 or 2.7 and some Python libraries²⁶:
 - ConfigParser²⁷
 - logging²⁸
 - gettext²⁹
- PyQt version 4.6 or later.
- Python for Windows Extensions³⁰ Build 214 or newer (needed for Microsoft Windows only).

It is recommended to use the default installation paths of the packages.

The next step is to configure tagstore before it is started the first time. For the purpose of this description, the installation folder of tagstore is denoted with \$bin. All command line examples and folder paths are written in posix format using forward slash (/) as the path separation character. Users of Microsoft Windows have to adapt the commands to the syntax of their system.

In the \$bin/tagstore/tsresources/conf/tagstore.cfg file, the user has to check or modify the settings for the following variables (which are described in Section 5.5):

- current_language (en or de)
- max_tags (depending on the system used; if unsure, retain the default value)
- first_start and all settings starting with show_ should be set to true
- The section below [stores] should be empty.

In the \$bin/tagstore/tsresources/conf/store.cfg.template file, the user has to set show_category_line which is described on page 126. This is the most important decision when setting up a store: whether or not to use a second tag line (for categories) and whether or not to use a [controlled vocabulary \(cv\)](#). She might as well check category_mandatory. All other

25. <http://www.python.org/getit/> – retrieved on 2012-08-18.

26. Some libraries are probably already installed on an average Python-equipped system.

27. <http://docs.python.org/library/configparser.html> – retrieved on 2012-08-18.

28. <http://docs.python.org/library/logging.html> – retrieved on 2012-08-18.

29. <http://docs.python.org/library/i18n.html> – retrieved on 2012-08-18.

30. <http://starship.python.net/crew/mhammond/win32/> – retrieved on 2012-08-18.

5 tagstore Implementation

settings could be changed using the tagstore Manager after the store has been set up.

If a cv is used and the user already has a list of vocabulary that should be re-used within tagstore, she can put this list – one word per line – in the `$bin/tagstore/tsresources/conf/store.tgs.template` file.

After setting up the configuration files, tagstore is ready to be started for the first time, setting up the first store. To manually start the tagstore Manager from the command line, the user changes her current path to `$bin/tagstore/` and invokes: `python ./tagstore_manager.py`

The following section continues with the description on how to set up the first store.

The tagstore implementation monitors the storage folders of each store. This can only be done when the tagstore software is running in the background. Installing tagstore using the installer packages automatically ensures that tagstore is started on system start. When using the manual installation process, the user has to set up such a starting command by herself. She can also choose to start tagstore manually by changing her current path to `$bin/tagstore/` and invoking: `python ./tagstore.py`

This tagstore background process only watches for changes in the storage folders of the stores. It has a very small memory footprint and does not need many CPU cycles. Hence, it does not slow down the system significantly.

Setting Up the First Store

The installation process should result in an open tagstore Manager GUI and an open assistant window. The latter helps the user through the process of creating a new store and is described in detail in Pirrer (2012). Since the location of a store can not be changed afterward, this decision should be made with care.

After the new store is created, the user is advised to go through all Manager tabs and check the settings. It is recommended to change settings like `datestamp`, `expiry date prefix`, and so on as soon as possible. This ensures a

consistent user experience. It is worth mentioning that some settings in the tagstore Manager are specific to stores. The tabs of those settings provide a drop-down menu on the top which should be used to select the desired store.

5.6.2 Updating the Installation

There is no automatic update process implemented yet. As long as the version number of the configuration files does not change, a manual update can be achieved by replacing the files in `$bin/tagstore/`.

It is important to back up (and later restore) the configuration settings in `$bin/tagstore/tsresources/conf/` in advance.

5.6.3 Adding Items

New items³¹ are added to a store by placing them into the storage folder of a store. This can be done by either using this path when saving new files within an application or by copying an existing item to this destination.

Once the tagstore background process recognizes a new file, the tagstore dialog window appears and the user is able to choose appropriate tags. The tag recommendation system (Section 5.3.1) and the tag completion feature make sure that the tagging effort is minimized as much as possible.

Multiple items can be added at once. The tagstore dialog will list all new items in its list. Tagging multiple items with the same set of tags is described in detail in Section 5.3.

³¹. It is important to know that a store is able to manage [files](#) as well as [folders](#). Placing folders in the storage folder is advised when working with software like \LaTeX (a \LaTeX document consists of \TeX files and several additional files) or any other self-contained folders like source code projects.

5 tagstore Implementation

5.6.4 Retrieving Items

The user is able to access items in a store along many different paths as shown in Figure 4.2 (p. 98).

Direct access Items in the storage folder are never modified by tagstore.³² Therefore, the user is able to re-find items by their names directly in the storage folder.

TagTrees The most important access method for items within tagstore is through **associative navigation** within the TagTrees. Within each store, there is a folder called **navigation**³³. In this folder, the user sees all the tags assigned to items, including the automatically assigned tags such as **datestamps** (described in Section 5.4.2).

Associative navigation is somewhat different to **navigation** within a **strict hierarchy** of folders and may require some time to become accustomed. Within TagTrees, the user chooses the first tag which seems to be appropriate. This is done by entering the folder whose name comprises the tag name.

Within this tag folder, the user is already able to locate *all* items that were tagged with this tag. In a populated store, the set of items in the first level of TagTrees might be very large. Therefore, the user might choose a second tag folder which might be related to the item she is looking for. Within this second level of the TagTrees, there are only items which were tagged using both, the first tag *and* the second tag. Thus, the set of items is most likely smaller than the set of items in the previous folder. These steps are repeated until the resulting set of items is small enough to scan easily.

One potential drawback is that an item has to be tagged by at least two tags to avoid the possible necessity to scan through a large set of items in

³². There is a single exception: expired items are *moved* to the `expired_items` folder as described in Section 5.4.3.

³³. For stores with two tag lines, there are two folders named `navigation` and `descriptions` instead.

the first level of TagTrees. The good thing is that none of the tag folders contain an empty set of items.

It is worth noting that TagTrees can be navigated within any “File open...” or “File save...” dialog from any application as well. Most other [PIM](#) applications do not offer this kind of system integration.

To sum up the possibilities, users are able to navigate and access items in the storage folder and all folders within TagTrees. Deleting and renaming items is restricted to the storage folder only. The next sections provide more information on this issue.

5.6.5 Renaming Items

If an already stored *file* is renamed in the storage folder, the tagstore background process is able to recognize this (unless multiple rename actions happen at the same time). Therefore, tagstore assigns the known set of tags from the old file name to the new file name without asking for new tags.

Renaming *folder* names results in a new tagging process. This is because tagstore is not able to determine similarities of folder content yet.

Any manual renaming within TagTrees is not recognized by tagstore and might result in broken links or errors. See Section [5.8.4](#) for more explanation.

5.6.6 Deleting Items

Items can be deleted from the storage folder any time. All related tag folders and links are removed from the TagTrees automatically as long as the tagstore background process is running.

Any manual delete action within TagTrees is not recognized by tagstore and might result in broken links or errors. See Section [5.8.4](#) for more explanation.

5.6.7 Getting Help

Any tagstore [GUI](#) offers context-specific help. These help windows can be opened by choosing the button with a question mark icon. Additional help is available in the [Frequently Asked Questions \(FAQ\)](#) on the project home page³⁴.

5.7 Best Practices

Several best practices and tips from long-term users have proven to be of value. Users of tagstore should be able to profit from the insights:

Easy-to-reach storage folder In order to get the best user experience, the user should be able to navigate to the storage folder very quickly. This can be achieved by either locating the store at an easy-to-reach location or linking the folder to an easy-to-reach location.

Particular applicability The tagstore software can be used for any kind of item. Many long-term users of tagstore reported particular applicability of tagstore for folders they previously named *misc*, *sundries*, or *download* folders of web browsers. Item types which demand multi-classification should be stored within a tagstore. This way, a user is able to classify a movie file like “*Delicatessen*”³⁵ with “*Fantasy*” and “*Comedy*”.

Desktop Search Engines There is no problem when tagstore is used on a system which runs a desktop search engine. I recommend any user to do so. To avoid multiple hits for any item stored in tagstore, the desktop engine should be configured to ignore all TagTrees. Items are indexed and found in the storage folders and the links from within TagTrees might irritate desktop search engines.

34. <http://tagstore.org> – retrieved on 2012-08-18.

35. <http://www.imdb.com/title/tt0101700/> – retrieved on 2012-08-19.

Backup software Any backup software can be used on a computer running tagstore. The user might want to add an exception rule for the TagTrees folders, if the backup software is not able to recognize [links](#) as links. In that case, the backup software replaces those links with copies of the corresponding target items. This results in unwanted multiplication of backup space.

Alternative file browsers Due to the design of tagstore, it can be used with any kind of file browser. Usually, the default file browsers (such as Windows Explorer or os x Finder) do not offer the best experience for advanced users. Alternative file browsers such as Free Commander³⁶ (Windows), Total Commander³⁷ (Windows), or Path Finder³⁸ (os x) allow much more advanced file navigation and management.

5.8 Technical Limitations

The theoretical number of links which would be created for each item is illustrated in Table 4.1 (p. 96). Additionally, a large set of folders has to be created for each tag in the TagTrees. This exponential relation between the number of tags per item and the number of links and folders has two main consequences in the current implementation of tagstore: using too many resources and taking too long to create.

5.8.1 Resources

[Inodes](#) are the smallest entities of file systems. Most file systems have a limited number of inodes that can be used on a single [partition](#). This means that the number of [files](#), [folders](#), and [links](#) is limited by this upper bound.

36. <http://www.freecommander.com/> – retrieved on 2012-08-19.

37. <http://www.ghisler.com/> – retrieved on 2012-08-19.

38. <http://cocoatech.com/pathfinder/> – retrieved on 2012-08-19.

5 tagstore Implementation

<i>tags/item</i>	<i>System 1 [s]</i>	<i>System 2 [s]</i>	<i>System 3 [s]</i>
4	0.02	0.35	0.46
5	0.08	1.26	2.39
6	0.38	7.55	14.09
7	4.13	51.06	104.75
8	88.87	421.05	798.04

Table 5.1: Performance measurements using three different combinations of hardware and software. The more tags are used, the longer it takes to create the TagTrees link structure. Up to four tags, any system is able to create TagTrees instantly. Using more tags per item, performance depends on the hardware and software used. System 1 runs GNU/Linux on a SSD, System 2 runs Windows 7 on a standard hard disk, and System 3 runs Windows 7 in a virtualized environment.

Usually, this number is so high, that most users never get near to this limit. However, heavily-linked structures like TagTrees can use up a large number of inodes very quickly. Eight new tags for a single item result in approximately 219,200 additionally used inodes.

Therefore, tagstore has a monitoring feature which reports the user when she is using up too many inodes per partition. A very rough estimation is that a store can hold a few thousand items without occupying all the inodes of an average partition.

5.8.2 Speed

The large number of folders and links which have to be created in the TagTrees when items are tagged, raises issues of speed. A small sample of performance measurements with the current version of tagstore is shown in Table 5.1. The more tags used, the longer it takes to create the TagTrees link structure. Up to four tags, any system is able to create TagTrees instantly. Using more tags per item, performance depends on the hardware and software used.

The systems which were tested to derive the measurements had the following properties:

- System 1:

5.8 Technical Limitations

- Hardware: lenovo X200s notebook, Samsung [SSD](#), intel Core 2 Duo CPU (L9400 1.86 GHz), four gigabytes of RAM
- Software: Ubuntu 11.04 GNU/Linux, 2.6.39 kernel (32bit)
- System 2:
 - Hardware: HDD, intel 6-core CPU (3.3 GHz), eight gigabytes of RAM
 - Software: Microsoft Windows 7 (64bit)
- System 3:
 - Hardware: Apple MacBook Air (2010), HDD, Intel Core 2 Duo (2.13 GHz), four gigabytes of RAM
 - Software: Windows 7 (32bit) being virtualized (on a OS X host) with 768 megabytes of virtual RAM and a virtual HDD

There are several issues which contribute to the differences between the three test systems. The most dominant ones are:

Windows shortcuts versus symbolic links Microsoft decided not to use some advanced features of [NTFS](#). One of them are [links](#) which are managed at the file system level³⁹. Windows uses [shortcuts](#) instead. Shortcuts are special kinds of files with the [file extension](#) “lnk”. The majority of end user applications can not handle links on [NTFS](#) correctly. This is the reason for tagstore to use shortcuts as well. Creating shortcuts is a user space process and requires much more than just telling the file system to create a symbolic link. That issue results in much worse link creation performance on Microsoft Windows systems.

SSD versus HDD Using a [Solid State Drive \(SSD\)](#) results in much better input/output performance than using a [Hard Disk Drive \(HDD\)](#). This has a direct impact on the performance of creating TagTrees.

The best performance can be achieved with a non-Windows operating system running on an SSD.

³⁹. On [NTFS](#) such links are called “junction points” (for folders) and “[NTFS symbolic links](#)” (for files).

5.8.3 Workarounds and Solutions

As long as the tagstore implementation does not use technologies like FUSE⁴⁰ or network drives, the limits on inodes and performance cannot be ignored. Some workarounds include:

Limiting the Number of Tags per Item

As a direct consequence of the fixed number of inodes and the speed issue, tagstore limits the maximum number of tags per item. The setting of `max_tags` (described on page 125) defines this setting. Users with the ability to edit configuration files manually are able to modify this setting according to their system environment.

Contrary to first thoughts, this does not represent a huge disadvantage. The average number of tags per item is small. For example, Hsieh et al. (2008, Figure 3) reported 3.1 tags per item on average and Pak, Pautz, and Iden (2007) 1.8 tags per item on average.

Choosing a Different File System

On GNU/Linux systems, the user can choose to use a file system which does not have these limitations on inode numbers. For example, [Extended File System \(xfs\)](#) or ReiserFS might not have this restriction.

Experienced GNU/Linux users are able to create a loop-back file system which is mounted on the path of the TagTrees folder. This way, the loop-back file system is used only for the TagTrees and represents a very clever solution to the limitation problem.

40. https://en.wikipedia.org/wiki/Filesystem_in_Userspace – retrieved on 2012-08-17.

5.8.4 No Adding, Deleting, or Renaming in TagTrees

The implementation of tagstore could theoretically allow items to be stored (or removed) directly in the TagTrees. Unfortunately, this requires tagstore to monitor all folders within the TagTrees hierarchy. At least Apple's os x limits the number of folders that can be monitored to 256. This number is exceeded very quickly. Therefore, tagstore is only able to recognize new, deleted, or renamed items within the storage folder as described in Section 5.6.5 and Section 5.6.6.

People using tagstore are able to store, delete, and rename items within the storage folder. Within the TagTrees, users are able to navigate to items using associations and open items.

5.9 Comparing tagstore to Other Solutions

There are a number of solutions on the market that address similar issues like tagstore does. In G. Binder (2012, in German) they were evaluated in detail.

- TagLauncher⁴¹
- TagEverything⁴²
- Tabbles⁴³
- tag2find⁴⁴
- TaggedFrog⁴⁵
- TaggTool⁴⁶
- WinDream
- Tagsistant⁴⁷

41. <http://www.taglauncher.com/> – retrieved on 2012-09-02.

42. <http://ww35.popularproductivity.com/tag-everything.html> – retrieved on 2012-09-02.

43. <http://tabbles.net/> – retrieved on 2012-09-02.

44. <http://www.tag2find.com/> – retrieved on 2012-09-02.

45. <http://lunarfrog.com/> – retrieved on 2012-09-02.

46. <http://www.taggtool.com/> – retrieved on 2012-09-02.

47. <http://www.tagsistant.net/> – retrieved on 2012-09-02.

5 tagstore Implementation

- Punakea⁴⁸
- Tags⁴⁹
- TagFS⁵⁰
- LabelFS⁵¹

48. <http://www.nudgenudge.eu/> – retrieved on 2012-09-02.

49. <http://www.caseapps.com/tags/> – retrieved on 2012-09-02.

50. <https://github.com/marook/tagfs/wiki/> – retrieved on 2012-09-02.

51. <http://code.google.com/p/labelfs/> – retrieved on 2012-09-02.

Did I disappoint you?
Or leave a bad taste in your mouth?

One
U2

6 Evaluation

The design and implementation of a new method was described in the previous chapters. But what about its effect on [PIM](#)? Are there positive or negative implications? How does this new method compare to people's current situation? This chapter gives a summary of the scientific evaluation of tagstore.

Long-term users provided valuable input not only for the development phase. Two formal experiments were conducted: the first one compared the traditional folder method to tagstore in the lab with a short time span between filing and re-finding tasks. The second formal experiment was similar to the first, one but with a two week pause between filing and re-finding tasks, more files, more test persons, and an enhanced test design.

The results of the experiments showed tremendously positive user feedback for tagstore. In general, objective performance measure were inconclusive. Thus, objective results and subjective feedback were contradictory in many issues. User acceptance, however, was very positive for tagstore.

6.1 Strategy

There are many different strategies to choose from when evaluating PIM methods or tools. Evaluation methods involving **test persons (TP)** can be classified in many different ways:¹

- temporal
 - short-term evaluation
 - long-term evaluation
- moderation style
 - moderated tasks test (typically: formal experiment)
 - co-discovery
 - unmoderated tasks tests
 - free-form tests
- level of interaction
 - interviews
 - questionnaires
 - doing a set of pre-defined tasks
 - using tool with own requirements (typically: long-term studies)
 - usage studies (typically: logging data)
 - A/B tests
- usage style
 - TPs silently doing tasks
 - TPs talking while doing tasks (thinking aloud tests)
- ...

In many cases, short-term evaluation is done by inviting **test persons (TP)** into a laboratory and asking them to do a set of pre-defined tasks. Such *formal experiments* have the advantage that the situation can be controlled to a maximum possible extent. I decided to perform laboratory experiments in order to evaluate several things:

1. Objectively compare tagstore to usual filing and navigation in **folder hierarchies**.

1. Dumas and Redish, 1999; Rubin and Chisnell, 2008; Mathis, 2011.

2. Obtain subjective feedback on the method and the implementation.
3. Detect usability issues in order to further optimize the implementation.

6.2 Informal Feedback from Long-Term Users

Starting at the development phase, several people volunteered to act as beta testers and long-term testers. It is very crucial to obtain long-term experience from outside the development team, as stated in Section 3.5.2. Direct feedback from long-term testers provided valuable input not only for the design phase, but also gave us insight into the practical implications. This feedback was used to form a collection of best practices (Section 5.7) and [FAQS](#), including:

Multimedia Content Users reported that tagstore seems to be an interesting method to manage multimedia content such as movies. Multi-classification for genre, actors, and so forth is important for these kinds of items.

Business and Sharing In the office, most people have to work on network drives whose structure is standardized in some more or less strict fashion. The implementation of tagstore is single-user, single-system and, therefore, not suitable for managing those kind of business items yet.

Misc, Sundries, ... Folders, which contain a composite mixture of all kinds of things, seem to be very suitable for tagstore. With multi-classification, a user is able to conquer this chaos.

Download Folders Download folders for browsers seem to be very suitable for tagstore. This is quite similar to the previous paragraph: very different kinds of files are stored there. Long-term users enjoy the possibility to browse through their downloads using associations.

6 Evaluation

Tags People using tagstore for a longer period of time report that their tagging vocabulary usually develops over the first weeks. After that, they restrict themselves to a smaller set of tags:

“ My number of tags did not grow any more: I use a tag set with about 20–30 tags, but the different combinations make it easy for re-finding files.

“ For me, tagstore was a new method for storing and finding files. It took me several months to really get into it. In the beginning, [I was using] only [the storage] folder, then really [started] with [the navigational structures of] tagstore. It is a new method, which can change the way of finding and re-finding files, but whether or not it will become a solution for everybody cannot be stated.

I am very sad, that on my business computer I currently have Windows 7 without administrator permissions. So for now I cannot use tagstore [on my business computer].

“ Different file types were used with several applications like image editors, music players or file browsers with tagstore without any problems. Furthermore, the additional dialog after saving or copying a file to the storage directory was not disturbing or very time consuming.

Looking for files in the folder hierarchies created by tagstore from tags shows its power. This means that it is very easy and time saving to re-find files. However, with the growing number of tags it could become a bit confusing because of the large number of directories at the top level of the descriptions or categories directory.

So it is worth saying that tagstore is a very big help for bringing different files in order. Due to that, it has been great for collecting files which are hard to classify in only one directory.

Integration One long-term user of tagstore who was running GNU/Linux expressed his desire that tagstore should be more integrated into his favorite file browser and web browser.

6.3 Formal Experiment 1

One of the most interesting things about a new method is how it performs when compared to the status quo. Therefore, a study which compares tagstore with the most frequently used method for filing and browsing items using Windows Explorer was designed. This direct comparison should yield both objective measures and subjective feedback.

The first formal experiment with tagstore took place in January 2011 and is described in detail in De Vocht et al. (2012). All relevant data concerning this experiment is published on github². The preliminary results of this study were published in Voit, Andrews, and Slany (2012b).

6.3.1 Methodology

This study was designed as a repeated measures (within-groups) formal experiment. A set of 18 German-speaking³ test persons (TP) was split into two random groups. Two conditions were tested: condition A was the Windows Explorer condition and condition B was the tagstore condition. Group 1 started with condition A first and did condition B afterwards. Group 2 started with condition B first and did condition A afterwards. This counterbalancing was done to compensate for learning or fatigue effects.

2. <https://github.com/novoid/2011-01-tagstore-formal-experiment> – retrieved on 2012-08-21.

3. Since the experiment took place in Graz/Austria, TPs spoke German. The tagstore software was configured to show a German interface. All user quotes in this document have been translated from German to English.

6 Evaluation

The tasks began with reviewing thirty test files⁴ on general topics. These test files were filed under both conditions and re-found under both conditions by every TP. Task times were measured, screen capture, user cam, and a room camera recorded the experiment. Sound recording was done as well.

One test user (not part of the 18 TPs) did a pilot test beforehand to test the experimental setup, tasks, equipment, process, and instructions.

All TPs were given fictitious identities (PT, TP01, TP02, . . . , TP18). Only one internal document contained the mapping of real names to the test IDs, to ensure protection of the volunteers' identities.

Each TP filled out a background questionnaire (see Figure 6.1 and Figure 6.2). During the experiment, performance data and completion rates were collected for each task. After finishing all tasks, an interview was conducted to gather spontaneous feedback. The first interview question was an open one ("How was it?") to encourage copious feedback. The final part consisted of filling out a feedback questionnaire (see Figure 6.3), where the TPs could express their opinion in a structured way.

Procedure

Group 1 had the following procedure:

1. The TP enters the test room.
2. The facilitator welcomes the TP and guides the TP to a table aside (not the one with the test computer).
3. The facilitator reads an orientation text which explains the rough procedure.
4. The TP is asked to fill out a background questionnaire form (see Figure 6.1 and Figure 6.2).
5. The TP is asked to sign the consent form.
6. The facilitator asks the TP to come to the table with the test computer.
7. The TP is asked to review the thirty test files (*task 1*).

4. The set of test files consisted of ten articles, ten files consisting of graphical content, and ten photographs.

6.3 Formal Experiment 1

8. The facilitator demonstrates filing of demonstration files into a new folder hierarchy.
9. The TP is asked to file the thirty test items into a folder hierarchy that has to be created by the TP (*task 2*).
10. The facilitator shows an instruction video⁵ which explains the basic usage of tagstore to the TP.
11. The TP is asked to file the thirty test items using tagstore (*task 3*).
12. The facilitator thanks the TP and the TP leaves the test room for a minimum of fifteen minutes to take a break.
13. The facilitator gets the TP back from the break and guides the TP to the table with the test computer.
14. The TP is asked to re-find six selected⁶ files (one by one, using Windows Explorer) within the folder hierarchy, which was made by the same TP in task 2 (*tasks 4–9*).
15. The TP is asked to re-find six selected files⁷ within the TagTrees folder structure of condition B using Windows Explorer (*tasks 10–15*).
16. The facilitator starts the interview phase by asking “How was it?”
 - In the interview phase, the facilitator tries not to steer the TP into any direction. If the TP stops talking, the following kinds of questions were asked: “Did you notice anything positive?” “Did you notice anything negative?” “Do you want to say something else?”
17. The facilitator ends the interview phase and guides the TP back to the other table.
18. The TP is asked to fill out the feedback questionnaire (Figure 6.3).
19. The facilitator thanks the TP and guides the TP to the door.

Group two had a similar procedure, only with the swapped conditions: *task 2* \leftrightarrow *task 3* and *tasks 4–9* \leftrightarrow *tasks 10–15*.

5. https://github.com/novoid/2011-01-tagstore-formal-experiment/tree/master/experiment_data/tagstore_introduction_video – retrieved on 2012-08-21.

6. The descriptions of three files were rather similar to the file name. The descriptions of the other three files were more vague.

7. The same set of selected files as in the previous six tasks for condition A.

6 Evaluation

Datum: 14.01.2011 Uhrzeit: 14:00 Test Nr.: 1 User Nr.: 5

Hintergrundbefragung

Danke, dass Sie sich als Freiwillige(r) für unseren Test zur Verfügung stellen.
Bitte beantworten Sie die folgenden Fragen:

1. Angaben zur Person

Geschlecht: ☒ männlich ☐ weiblich

Alter: 39

Beruf: Produkt Entwickler

2. Sehvermögen

1. Verwenden Sie eine Sehhilfe bei der Arbeit am Computer?

☒ Keine ☐ Brille ☐ Kontaktlinsen ☐ Sonstige _____

2. Sind Sie farbenblind?

☒ Nein ☐ Ja, und zwar _____

3. Ausbildung

1. Abgeschlossene Ausbildung:

☐ Lehre ☒ Matura ☐ Studium ☐ Doktorat

2. Wenn Sie studieren oder studiert haben, beschreiben Sie bitte Ihr Hauptstudiengebiet:

4. Umgang mit Computern

1. Wie lange benutzen Sie bereits Personal Computer?

25 Jahre

2. Wieviele Stunden pro Woche verwenden Sie einen Computer?

60 Stunden

3. Welche Art von Computer verwenden Sie am meisten?

☒ Microsoft Windows ☐ Apple Macintosh ☒ Unix
☐ Sonstige _____

5. Domain-Spezifische Fragen

1. Welchen Datei-Browser verwenden Sie hauptsächlich?

☒ Windows Explorer ☐ Mac Finder ☐ Weiß nicht
☐ Sonstiger Neutler

2. Ist Ihnen der Begriff "Tagging" bekannt?

☒ Ja ☐ Nein

3. Wenn ja, was verstehen Sie darunter?

Schlagwortvorgebe

Figure 6.1: An example background questionnaire from TP05, page one. The header information was filled out by the facilitator, the questionnaire was filled out by the test persons.

6.3 Formal Experiment 1

4. Verwenden Sie selbst Tagging?	<input checked="" type="checkbox"/> Ja	<input type="checkbox"/> Nein	
5. Wenn ja, wo verwenden Sie Tagging am häufigsten?	Bloggen		
6. Wie viele Emails befinden sich für gewöhnlich in Ihrem Posteingang?	<input type="checkbox"/> < 5	<input checked="" type="checkbox"/> 5-40	<input type="checkbox"/> > 40
7. Haben Sie Ihre eigene hierarchische Struktur in der Sie Ihre Emails archivieren, oder lassen Sie alle Emails im Posteingang?	<input checked="" type="checkbox"/> Hierarchische Struktur	<input type="checkbox"/> Posteingang	
6. Erfahrung mit Usability Tests			
1. Haben Sie schon an eine Usability Studie teilgenommen?	<input type="checkbox"/> Ja, als Testperson		<input type="checkbox"/> Ja, als Mitglied des Testteams
	<input checked="" type="checkbox"/> Nein		
Wenn ja, was war das für eine Studie?			_____

Figure 6.2: An example background questionnaire from TP05, page two. Originally, this was on the back side of part one.

Test Persons

The TPs were recruited on various occasions. I gave many talks about my research work at business meetings, and BarCamps⁸ in Graz⁹ and Vienna¹⁰. Using these events and modern social media I was able to encourage people to send me an email and fill out a short online survey for basic information about their computer background. So far, I received emails from 168 interested German-speaking persons¹¹ and 116 persons¹² from all over the world.¹³

Together with friends and colleagues, I was able to recruit 18 persons (15 male, 3 female) for the two days of testing. No test person volunteering for the experiment received any payment.

8. <https://en.wikipedia.org/wiki/BarCamp> – retrieved on 2012-08-21.

9. <http://barcamp-graz.at/> – retrieved on 2012-08-21.

10. http://www.barcamp.at/UxCamp_2010 – retrieved on 2012-08-21.

11. 128 persons have completed the online questionnaire.

12. 60 persons have completed the online questionnaire.

13. The numbers were derived on 2012-08-22.

6 Evaluation

Datum: 14.01.2011 Uhrzeit: 14:55 Test Nr.: 1 User Nr.: 5

Feedback Formular

Bewerten Sie bitte anhand folgender Aspekte Ihre Vorlieben für die eine oder die andere Dateimanagement-Variante.

1.	Wie gut gefiel Ihnen die Ordner-Variante ohne tagstore beim Ablegen?	Sehr gut	3	2	1	0	<input checked="" type="checkbox"/>	2	3	Sehr schlecht
2.	Wie gut gefiel Ihnen tagstore beim Ablegen?	Sehr gut	3	2	<input checked="" type="checkbox"/>	0	1	2	3	Sehr schlecht
3.	Wie gut gefiel Ihnen die Ordner-Variante ohne tagstore beim Wiederfinden?	Sehr gut	3	2	1	0	<input checked="" type="checkbox"/>	2	3	Sehr schlecht
4.	Wie gut gefiel Ihnen tagstore beim Wiederfinden?	Sehr gut	3	2	<input checked="" type="checkbox"/>	0	1	2	3	Sehr schlecht
5.	Als wie schnell empfanden Sie die Ordner-Variante ohne tagstore beim Ablegen?	Sehr schnell	3	<input checked="" type="checkbox"/>	1	0	1	2	3	Sehr langsam
6.	Als wie schnell empfanden Sie tagstore beim Ablegen?	Sehr schnell	3	2	1	0	<input checked="" type="checkbox"/>	2	3	Sehr langsam
7.	Als wie schnell empfanden Sie die Ordner-Variante ohne tagstore beim Wiederfinden?	Sehr schnell	3	2	<input checked="" type="checkbox"/>	0	1	2	3	Sehr langsam
8.	Als wie schnell empfanden Sie tagstore beim Wiederfinden?	Sehr schnell	3	2	1	0	<input checked="" type="checkbox"/>	2	3	Sehr langsam
9.	Wie gut unterstützte Sie die Ordner-Variante ohne tagstore beim Ablegen/Wiederfinden von Bildern?	Sehr gut	3	2	1	<input checked="" type="checkbox"/>	1	2	3	Sehr schlecht
10.	Wie gut unterstützte Sie die Ordner-Variante ohne tagstore beim Ablegen/Wiederfinden von Textdokumenten?	Sehr gut	3	2	<input checked="" type="checkbox"/>	0	1	2	3	Sehr schlecht
11.	Wie gut unterstützte Sie tagstore beim Ablegen/Wiederfinden von Bildern?	Sehr gut	3	<input checked="" type="checkbox"/>	1	0	1	2	3	Sehr schlecht
12.	Wie gut unterstützte Sie tagstore beim Ablegen/Wiederfinden von Textdokumenten?	Sehr gut	3	<input checked="" type="checkbox"/>	1	0	1	2	3	Sehr schlecht
13.	Alles in allem, welche Variante würden sie bevorzugen?	[] Ordner <input checked="" type="checkbox"/> Tags								
	Weswegen?	<u>endlich - cool!</u>								
14.	Könnten Sie sich vorstellen, tagstore auf Ihrem Rechner einzusetzen?	Ja, auf jeden Fall	<input checked="" type="checkbox"/>	2	1	0	1	2	3	Nein, niemals
	Weswegen?	<u>logische Ordnung</u>								

Figure 6.3: An example feedback form from TP05. The header information was filled out by the facilitator, the rest of the form was filled out by the test persons.

6.3 Formal Experiment 1



Figure 6.4: The test environment with the observing person on the left-hand side (operating logging devices, room camera, and further equipment), the test person working on the test computer, and the facilitator on the right-hand side. In the foreground, there is the table for filling out questionnaires and feedback forms.

The age of the TPs ranged from 21 to 40 with a median of 28. The average amount of computer experience was 16.28 years. The average computer usage time per week was 45.22 (according to the answers to the background questionnaire). Most of the TPs were highly educated: only two of them did not have a university degree. Half of the TPs did not use tagging methods before.

Test Environment

Room In the test room there were only the TP, the facilitator and one or two persons handling the recording equipment. Outside the test room, there was an additional person who took care of the TPs arriving, waiting, or leaving. Inside and outside the room, the TPs were offered non-alcoholic

6 Evaluation

beverages and snack food. Besides the computer and recording equipment, there were three tables: observer/recording, table for filling out questionnaires/feedback, and the test computer table.

Hardware The test computer was a Sony Vaio Core 2 Duo notebook with a display resolution of 1280×800 on 13.3 inches. Video recording was done using a digital camcorder from Sanyo (Xacti HD1010). The observing person was using a notebook for taking notes and had a standard TFT display which showed the mirrored display content from the test notebook.

Software For recording the sessions, Morae v3.2.1¹⁴ was used: Morae Recorder on the test computer, Morae Observer on the computer of the observing person, and Morae Manager to post-process the recordings. The test computer was running an English¹⁵ version of Windows 7 (32bit) with Internet Explorer 7.0. For the experiment, tagstore¹⁶ was used with one tag line and no **controlled vocabulary**.

Limitations and software issues Due to the development stage of tagstore, this version lacked some features which are described in Chapter 5. There was no tag cloud, no advanced recommendation system, no online help system, and no set-up assistant. Instead of the advanced recommendation system, this version had a simple recommender, which showed a set of tags directly below the tag line, as shown in Figure 6.5. The suggestions were clickable and derived from the most recently and the most often used tags of the user. Unfortunately, a bug was discovered in the software: when a file containing a German umlaut¹⁷ was added to tagstore, this item re-appeared in the list of untagged items. Another bug in the tagstore software caused the background process to crash when the tagging dialog was closed using the red “x” icon in the upper right position. All time ranges

14. <http://www.techsmith.com/morae.html> – retrieved on 2012-08-22.

15. The test laptop with Morae Recorder installed was running an *English* version of Windows. Our German-speaking tps did not show any particular issues because of the different language.

16. The tagstore software used was Rev. 226 from 2011-01-13T15:12.

17. German umlauts are: ä, Ä, ö, Ö, ü, and Ü.

6.3 Formal Experiment 1

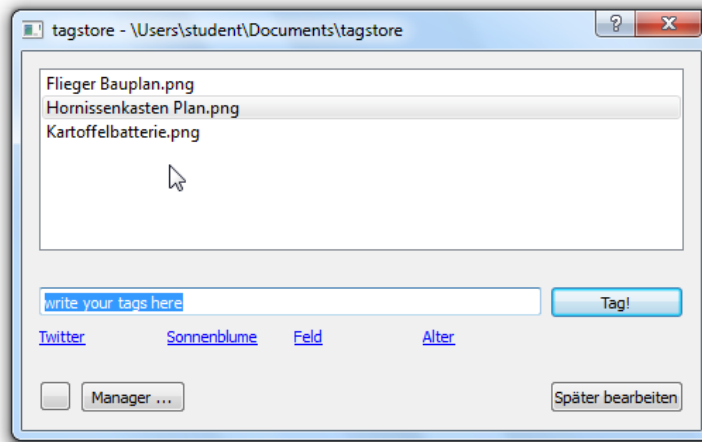


Figure 6.5: A screenshot of tagstore rev.226 showing three untagged files, an empty tag line, and four tag recommendations.

where bugs distracted the attention of the TPs were removed from the resulting data set (distractions).

Training Since all TPs were familiar with the Windows operating system, using Windows Explorer should not lead to any problem. Nevertheless, the facilitator demonstrated the task of selecting files and creating a folder hierarchy to file those items. No TP was familiar with the tagstore software prior to the experiment. A training screencast video¹⁸ of tagstore was created, showing the basic principles and features. This way, all TPs learned tagstore in the exact same fashion, not favoring any TP by different explanation. After the training video, adding demo files to the installed tagstore instance was shown by the facilitator. The set of demo files differed from the test files completely.

Data Preparation Each TP had the following situation at the beginning of the session:

18. https://github.com/novoid/2011-01-tagstore-formal-experiment/tree/master/experiment_data/tagstore_introduction_video – retrieved on 2012-08-21.

6 Evaluation

- Windows 7 (English) with an external standard computer mouse at the side preferred by the TP.
- A folder on drive letter M: containing three demonstration files which were used by the facilitator to demonstrate how to file in a folder hierarchy or tagstore.
- An empty drive with letter O:, where the TPs had to create their folder hierarchy¹⁹.
- On drive letter T: there was an empty tagstore store²⁰.
- An open Windows Explorer window showing drive letter M: containing thirty test files²¹ as shown in Figure 6.6.

On each new starting session, several scripts prepared the situation described above. When a TP finished the first session (tasks 1–3) the current state of the folder drive and the tagstore drive were archived using scripts. Before a TP returned from the session break, the corresponding situation from the previous session was restored using scripts.

Data Processing

The logs from the Morae software were not usable for the purpose of evaluation. On the one hand, TPs did things we could not pre-define within the Morae configuration, so the Morae summaries for task times were not correct. And on the other hand, the person on the observing interface was not able to follow short-term actions of the TPs in time. That resulted in time logs which were off by a high percentage of the actual task time.

Therefore, for post-processing the video logs of the sessions, a new approach was developed. The team had to analyze the screencast videos from the tasks in detail. Each spoken word was transcribed into text files. For task action and time logs I developed [deterministic finite automats \(DFA\)](#) and a simple logging language. The DFAs mapped the most important states of the TPs.

¹⁹. The letter “O” was chosen because “folder” translates to the word “Ordner” in German.

²⁰. tagstore was configured to show one tag line and no [CV](#) (“My Tags”).

²¹. The set of test files consisted of ten articles, ten files consisting of graphical content, and ten photographs.

6.3 Formal Experiment 1

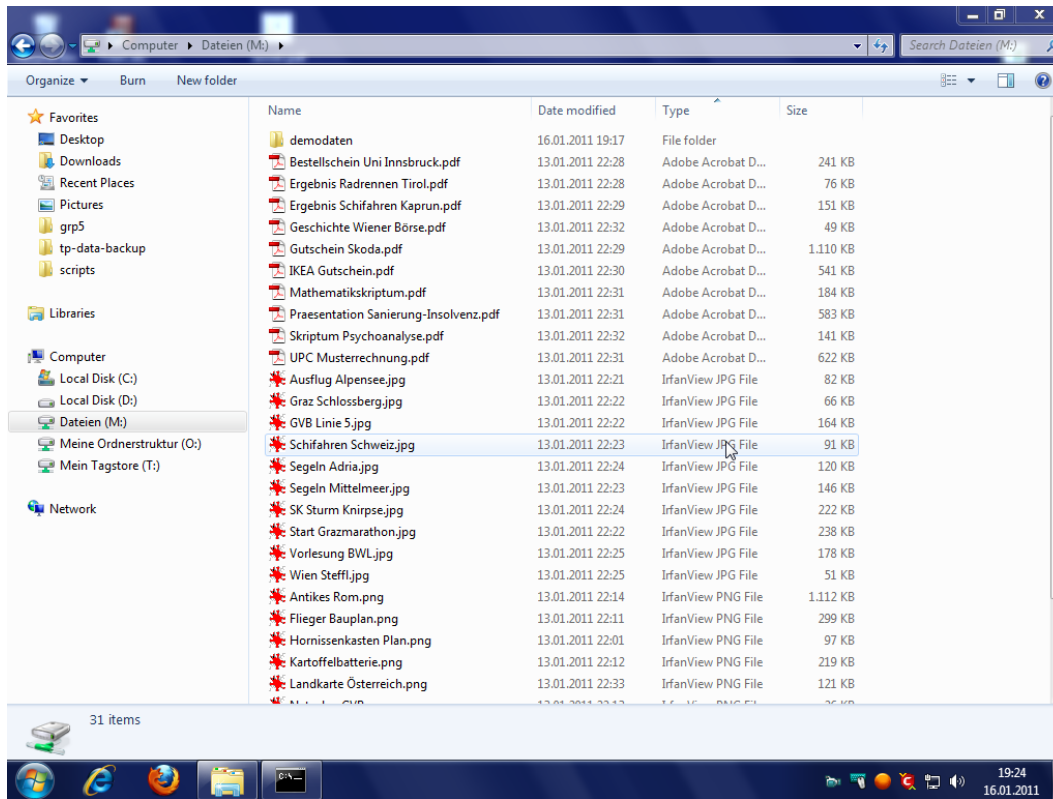


Figure 6.6: Windows Explorer showing the test files and the mapped drives for test files (M:), the folder hierarchy (O:), and tagstore (T:).

The video logs were written in text files that contained one line per log entry. Each line started with a time stamp which consisted of the absolute minute and second within the video file. Following the time stamp, an event shortcut described what happened. Depending on the event shortcut name, a set of parameters had to be added. Some event shortcuts did not require any parameter and some event shortcuts had optional parameters at the end of the line.

Figure 6.7 visualizes the data processing from the manual video transcript, to the event shortcut log, to the parser which derives result files, and separate script files deriving additional result files.

6 Evaluation

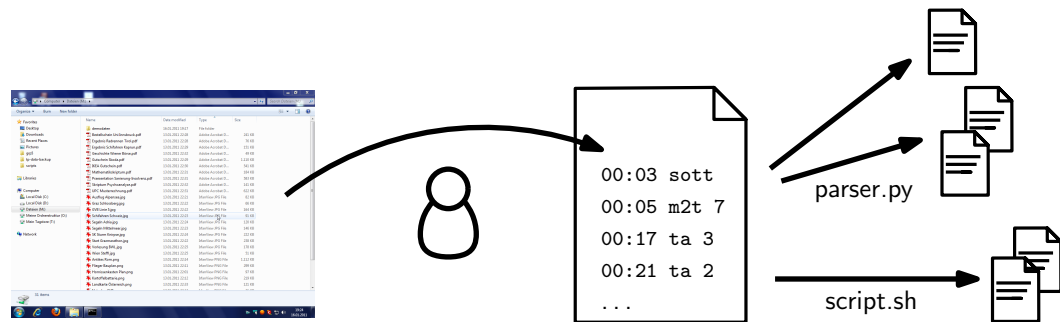


Figure 6.7: Data processing: Manual transcriptions of the video screencasts resulted in the event shortcut log files. Dedicated parsers processed these log files and generated a number of csv result files. Shell scripts derived additional result files.

For filing in folders, the states of the DFA were:

start Starting state before the first TP action. When a TP interacted with the computer for the first time, the task time for the next state started.

picking The TP selected files which were added to the folder hierarchy.

filing The TP managed the folder hierarchy and filed the selected test files.

end When a TP decided that the task was finished, logging of the task time stopped.

Figure 6.8 shows the DFA for the folder-filing task of the first formal experiment. It contains the states as circles and the event shortcuts as transition rules between states.

sotf ... start of task: filing in folders

Example: 5:55 sotf

o ... changing current view to drive 0: (folder)

Example: 5:55 o

m2o *n* ... move *n* selected files from M: to 0:\

Example: 5:59 m2o 3

m2f *n folder* ... move *n* selected files from M: to a sub-folder on 0:

Example: 6:05 m2f 2 Business

m ... change current view to drive M: (test files)

Example: 6:09

6.3 Formal Experiment 1

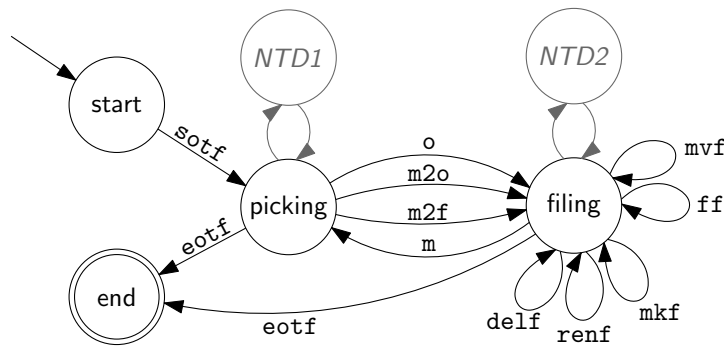


Figure 6.8: The deterministic finite automaton (DFA) for the filing into a folder hierarchy task for the first formal experiment. The TPs started at the file-picking state (selecting test files), switched to the filing state (creating and managing the folder hierarchy) and back to the picking state or end state. For simplicity, the two states which start with “NTD” summarize the non-task distractions and are shown in detail in Figure 6.10.

`mvf n dest ... move n folders into the folder named dest`

Example: 6:12 `mvf 3 "sports leisure"`

`ff n dest ... move n files from a sub-folder on drive 0: to the dest folder on drive 0:`

Example: 6:19 `ff 4 vacation`

`delf folder ... delete folder named folder`

Example: 7:14 `delf "education"`

`renf from to ... rename the folder from to to`

Example: 7:20 `renf university education`

`mkf folder ... create the folder named folder`

Example: 7:22 `mkf Austria`

`eotf ... end of task: filing in folders`

Example: 8:11 `eotf`

For tagging with tagstore, the states of the DFA were:

start The starting state before the first TP action. When a TP interacted with the computer for the first time, the task time for the next state started.

picking The TP selected files which were added to the store.

tagging The TP was facing the tagstore dialog.

6 Evaluation

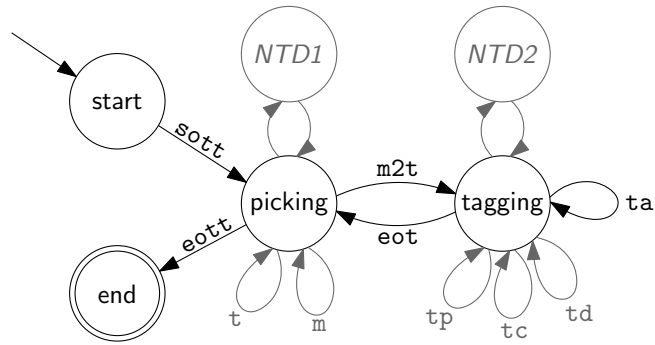


Figure 6.9: The deterministic finite automaton (DFA) for the tagging into a folder hierarchy task for the first formal experiment. The TPs started with the file-picking state (selecting test files), switched to the tagging state (adding files to tagstore) and back to picking state. For simplicity, the two states which start with “NTD” summarize the non-task distractions and are shown in detail in Figure 6.10.

end When the last tagstore dialog window closed, the task was finished, logging of the task time stopped.

Figure 6.9 shows the DFA for the tagging task using tagstore. It contains the states as circles and the event shortcuts as transition rules between states.

sott ... start of task: filing in tagstore

Example: 14:12 sott

m2t n ... move n files from M: (test files) to T:\tagstore\Ablage²²

Example: 14:22 m2t 5

ta n ... assign n tags to an item²³

Example: 14:27 ta 3

tp ... usage of tag which was proposed by the recommender

Example: 14:28 tp

tc ... usage of the tag completion feature

Example: 14:31 tc

td ... TP used at least one default tag²⁴

Example: 14:33 td

²². Ablage is the name of the storage folder in a German tagstore.

²³. Tagging of multiple items was not implemented in this tagstore version.

²⁴. Default tags are the tags from the previous item.

6.3 Formal Experiment 1

eot ... end of tagging; the tagstore dialog closes
Example: 14:34 eot
t ... change to drive T: (tagstore)
Example: 14:36 t
m ... change to drive M: (test items)
Example: 14:41 m
eott ... end of the task; when the last item was tagged, the task time stopped
Example: 14:59 eott

Distraction event shortcuts were used in all filing sessions (folders and tagstore):

ib ... begin inspecting file(s) by the TP
Example: 10:02 ib
ie ... end inspecting file(s)
Example: 10:07 ie
ct *comment* ... begin of a comment by the TP
Example: 10:09 ct "Where am I now?"
cf *comment* ... begin of a comment by the facilitator
Example: 10:12 cf "Please ignore this bug."
cl *comment* ... begin of a comment by the person who was logging
Example: 10:16 cl "Please start the screencast recording"
ce ... end of the previously started comment (by anyone)
Example: 10:28 ce
fb *comment* ... facilitator takes over control (due to technical issue)
Example: 10:48 fb "tagstore but with umlaut"
fe ... end of facilitator taking over control
Example: 10:59 fe

The re-finding process was much simpler. Since the tasks were rather short, we ignored all distractions that might have occurred in between. The facilitator asked for a file and when the test person started to interact with the computer, the task time was started. When the correct file was opened and its content was visible on the screen, the task time was stopped. Figure 6.11 shows the DFA of the re-finding tasks.

tf *taskn time* ... task number *taskn* finished successfully within *time*

6 Evaluation

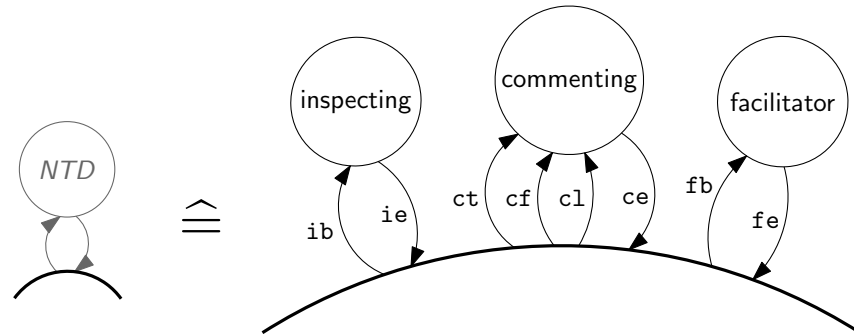


Figure 6.10: Distractions, which were not related to accomplishing the tasks, were summarized with the term “NTD”. We had three different distraction states: “inspecting” (TPs opening files and re-inspecting the content), “commenting” (TP, facilitator, or logging person speaking), and “facilitator” (facilitator took over control of the computer; mostly for fixing technical issues). In order to be able to compare the methods, all distraction times (based on exact measures of seconds) were removed from the result set.

seconds (in deciseconds, a tenth of a second)

Example: 1:05 tf 5 2.4

tc *taskn* *time* *comment* ... task number *taskn* was cancelled after *time* with a *comment*

Example: 3:06 tc 7 28.3 "TP gave up"

The extraction of those detailed logs was very time-consuming. The logs were separated in different files: one file for the filing task of the folder condition, one file for the filing task in the tagstore condition, one file for

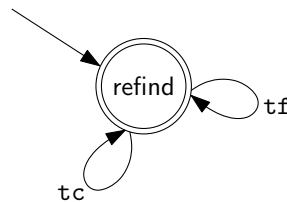


Figure 6.11: The DFA for refinding is a very simple one because only (short) task completion times and task cancel events were extracted.

6.3 Formal Experiment 1

re-finding within folder condition, and one file for re-finding within tagstore.

A Python compiler was developed which parsed the log files and derived files in `csv` format. The compiler also computed absolute task times without distractions.

All logging files, the derived `csv` files, and the parser script were published on the github repository.²⁵

6.3.2 Results

As mentioned in the previous section, all raw data files, parsers, scripts, and result files are available online.²⁵ In the following section, all references to item names in the git repository are denoted with footnotes like this²⁶.

Unless otherwise stated, the following applies to the statistical results:

- All distraction times (as described in the previous section) were removed from the data set.
- A “paired *t*-test” was used to determine statistical significance (Lewis and Sauro, 2012, Chapter 5, example 2).
- A confidence interval of 0.95 was used.
- Details of the statistical results can be found in the git repository.²⁷

Success Rate

The success rate was 100 percent²⁸. There was one exception, where it was not possible to re-find items because of a technical issue. These two incidents were removed from the evaluation results, because they were not re-finding issues of the TP.

²⁵. <https://github.com/novoid/2011-01-tagstore-formal-experiment> – retrieved on 2012-08-22.

²⁶. `git: an.example_file.txt`.

²⁷. `git: results.org`.

²⁸. `git: refinding_folders.csv refinding_tagstore.csv`.

6 Evaluation

	Measure	Folders	tagstore
Filing	Time (all)	451.35 \pm 35.78	542.47 \pm 58.14
	Time (fast-perf.)	385.75 \pm 20.93	337.38 \pm 13.59
Re-finding	Clicks (all)	3.17 \pm 0.50	2.32 \pm 0.26
	Time (all)	25.05 \pm 1.58	26.03 \pm 1.85
	Time (fast-perf.)	34.38 \pm 3.83	33.75 \pm 1.98

Table 6.1: FE1: Filing times, re-finding times and re-finding mouse clicks with their mean values and standard errors. Mean values show that filing in folders was faster, whereas re-finding performance did not vary between the conditions. The number of mouse clicks when re-finding showed a significant difference with $p < 0.005$ ($t(16) = 3.31$).

Task Performance

Table 6.1 shows a summary of the task times for filing and re-finding for both conditions.

Filing in folder condition was faster than filing in tagstore condition, although not statistically significantly with $p > 0.11$ ($t(16) = -1.69$). Figure 6.12 shows a bar chart of the filing performance of all tps and Figure 6.13 visualizes the same data in a box-and-whisker diagram (in short: boxplot).

Re-finding in both conditions was very similar with $p > 0.91$ ($t(16) = -0.11$).

In general, re-finding the first file took longer than the other five files.²⁹ However, for re-finding of file two to six (Figure 6.14, Figure 6.15) there is still a statistically significant difference ($p \cong 0.5$ with $t(16) = -0.63$).

The number of mouse clicks³⁰ used to re-find files differs with $p < 0.005$ ($t(16) = 3.31$, Table 6.1). Re-finding with tagstore requires statistically significant less mouse clicks.

In Figure 6.16, there is a clearly visible gap between tp 3 and tp 15. The set of tps with faster performance compared to tp 3 is also the group of the

²⁹. git: results.ods.

³⁰. git: results.ods.

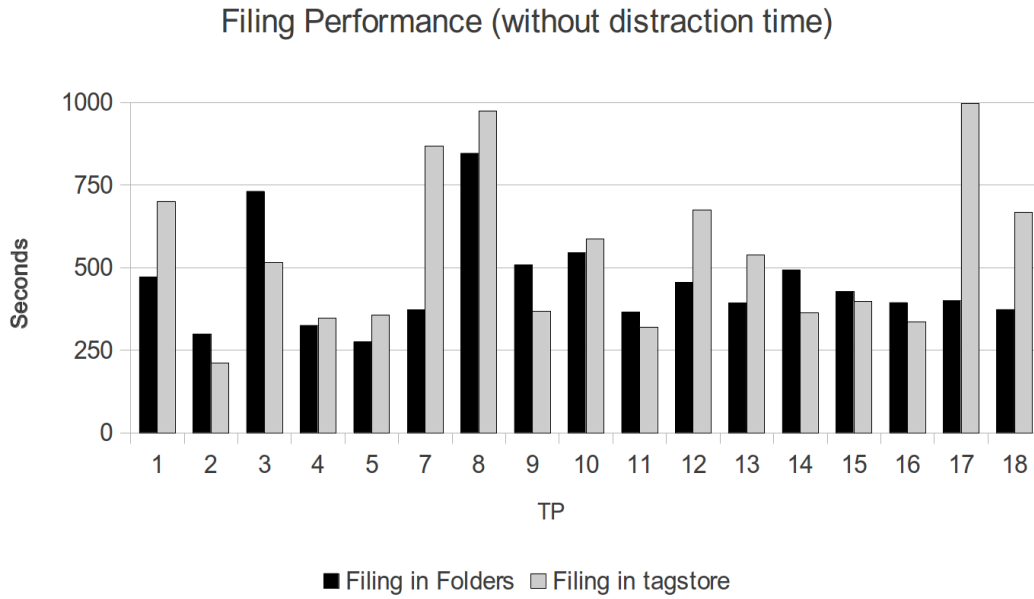


Figure 6.12: Filing performance for both conditions. The large standard deviations are visible in the variations between the TPs. There is also no clearly visible advantage for one condition over the other.

nine fastest performing TPs in the folder condition. This sub-group of TP 2, 4, 5, 9, 11, 14, 15, 16 is equally divided between the two groups. Hence, this sub-group is a valid sub-group for further analysis. This set of TPs is called “fast performers”.

Within the fast performer group, filing performance in the two conditions is the other way around: filing in tagstore is faster within the fast performers group in absolute numbers. But this is not statistically significant with $p \approx 0.11$ ($t(7) = 1.83$). The p -value is similar to the one from filing by all TPs, but favors the other condition. Figure 6.17(a) shows the bar chart of the filing performance of the fast performers.

For re-finding files, there is no difference to the set of all TPs: with $p > 0.90$ ($t(7) = 0.13$) the re-finding performance does not differ between conditions, as visualized in Figure 6.17(b).

6 Evaluation

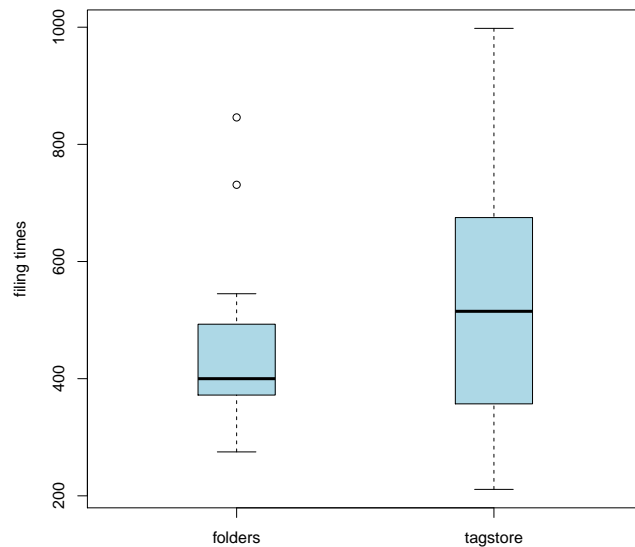


Figure 6.13: A boxplot of the filing tasks: in general, the folder condition was faster compared to the tagstore condition for filing, but not statistically significant standard deviation. The absolute times for the tagstore condition varied to a greater extent.

Artifacts

During the tasks, each TP created two sets of artifacts: the folder hierarchy containing the test files and the status of tagstore after tagging the test files.

On average, a TP used 9.1 characters per folder name³¹. Each test file is located within one or more folders that form the path to this file. The number of (parent) folders to this file is a similar measure to the number of tags associated to a file in tagstore. The average number of folders to a file is 1.7.³² When filing in tagstore, a TP used 2.7 tags per file on average³³ with an average tag length³⁴ of 7.6.

³¹. git: additional_statistics.org.

³². git: folders_combined_list_of_average_nr_of_parentfolders_per_TP_-_average.csv.

³³. git: average_nr_of_tags_-_general_value.csv.

³⁴. git: additional_statistics.org.

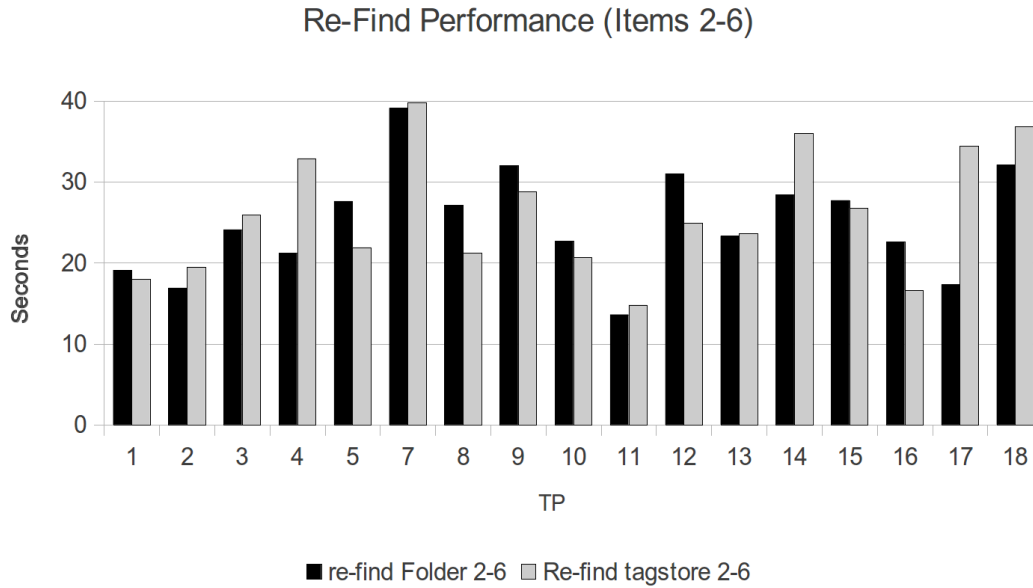


Figure 6.14: Re-finding performance for both conditions for file 2–6.

Figure 6.18 shows a box-and-whisker diagram (in short: boxplot) which compares the list of number of unique folders created by the TPs³⁵ and the number of unique tags of the TPs³⁶. It is obvious that the TPs used far less different folder names than different tag names.

When comparing the average length of folder names³⁷ to the average length of tags³⁸ in Figure 6.19, the difference of the average categorization strings is visualized in more detail: tags tend to be slightly shorter than folder names.

The average number of (parent) folders to a file³⁹ compared to the average number of tags per file⁴⁰ is a measure on how many categories are “as-

³⁵. git: folders_number_of_unique_per_TP.csv.

³⁶. git: tags_number_of_unique_per_TP.csv.

³⁷. git: folders_without_path_-_lengths.csv.

³⁸. git: tags_concatated_-_raw_tags_single_unique_-_lengths.csv.

³⁹. git: folders_list_of_average_nr_of_parentfolders_per_TP.csv.

⁴⁰. git: average_nr_of_tags_-_list.csv.

6 Evaluation

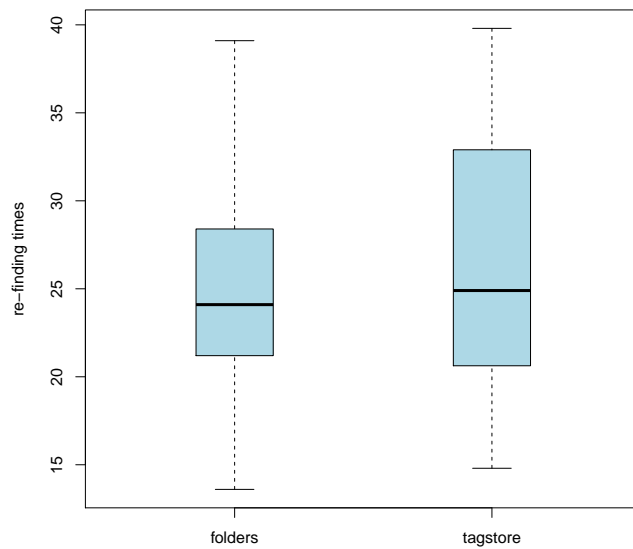


Figure 6.15: A boxplot of the re-finding tasks: this is a comparison of the re-finding times in the folder condition to the tagstore condition reduced to file 2–6. The performances were very similar in both conditions.

sociated” to a file. Figure 6.20 shows that in general there are more tags associated to a file than (parent) folders to a file.

Feature Usage

The detailed transcriptions contain feature usage information. As a rough measurement, the number of TPS that used convenience features are listed in Table 6.2.

From these three features, the *default tag* feature and the *tag completion* feature were used by the majority of TPS. The tag recommendation feature was used by five TPS. The fast performers group did not differ in a clear way: they were using these features in a similar way to the non-fast performing TPS.

6.3 Formal Experiment 1

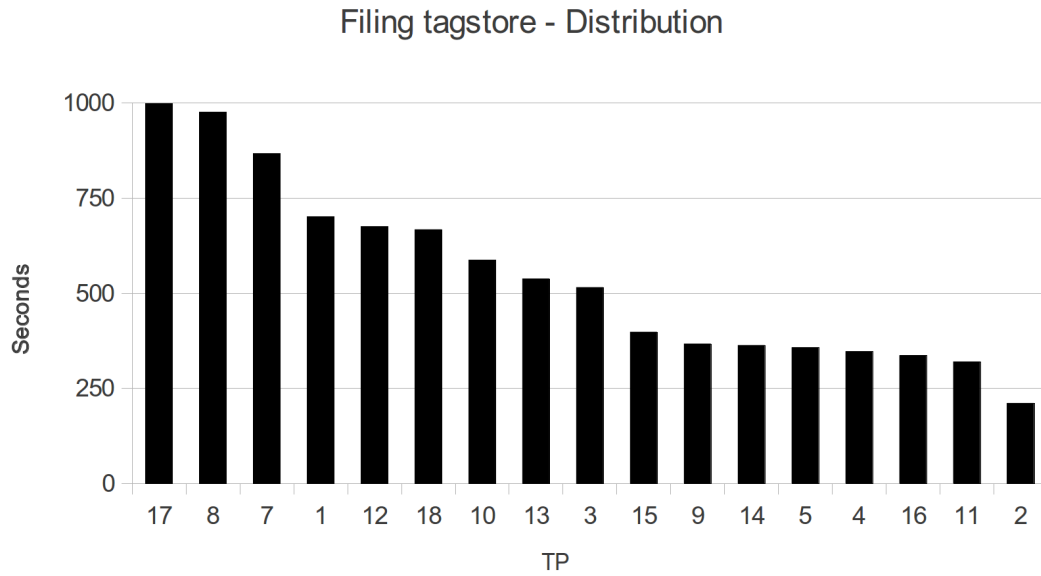
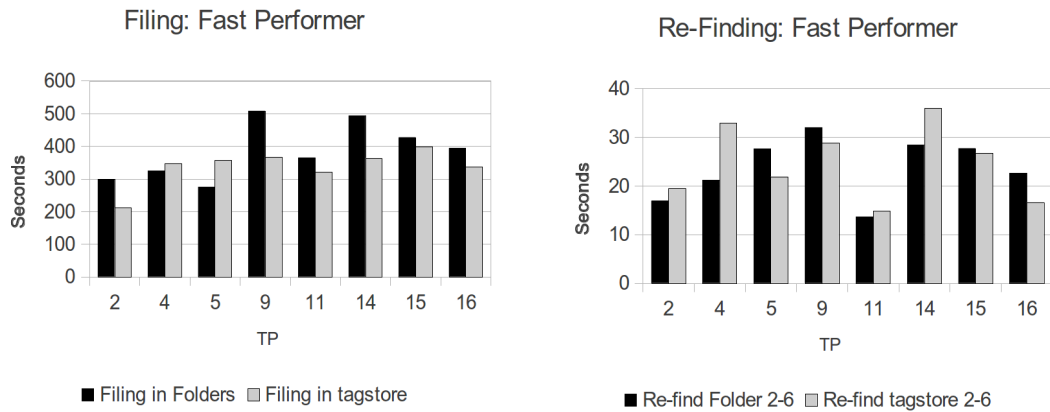


Figure 6.16: Filing performance within tagstore sorted by descending performance. There is a gap between TP 3 and TP 15. The sub-group of TP 15 to 2 is called “fast performers”.



(a) Filing performance of the fast performers group: the absolute numbers show that filing was faster in the tagstore condition but this was not statistically significant.

(b) Re-finding performance of the fast performers group. No significant difference could be found.

Figure 6.17: Filing and re-finding performance of the fast performers group.

6 Evaluation

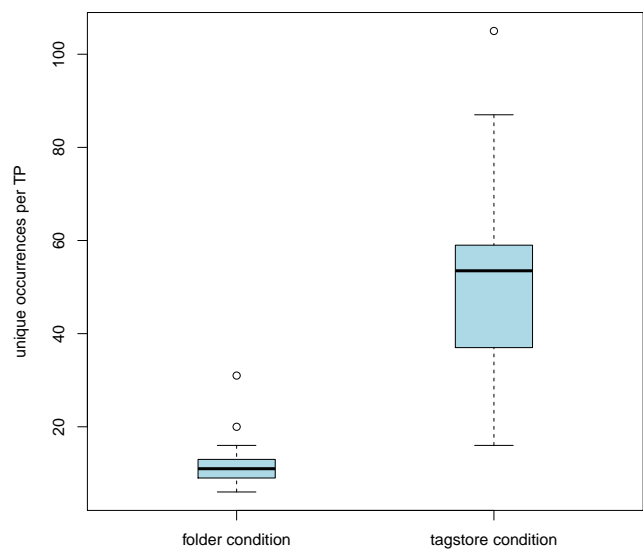


Figure 6.18: A boxplot comparing the number of (unique for each TP) folders with the list of tags (unique for each TP) of the TPs. There were many more different tags used than different folder names created.

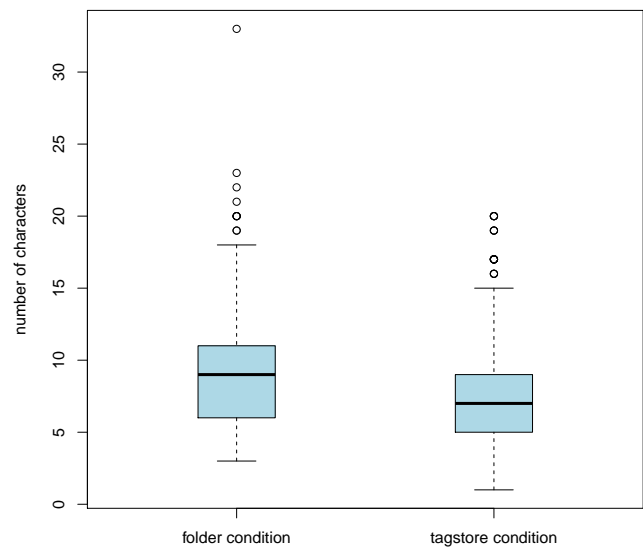


Figure 6.19: A comparison of the lengths of folder names and tag names. Tag names are slightly shorter than folder names.

6.3 Formal Experiment 1

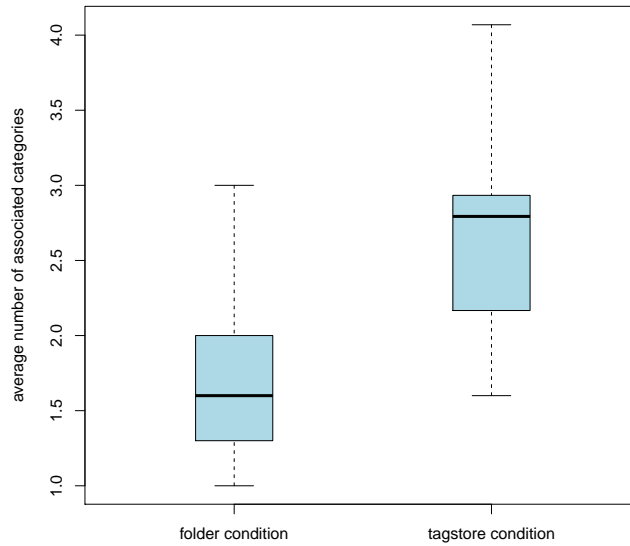


Figure 6.20: The number of associated (parent) folders of a file-path compared to the number of tags attached to files shows that there are more tags associated to a file than there are folders.

Feature	Nr. of TPS	List of TPS
default tags	14	1, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 18
tag completion	12	1, 3, 4, 5, 7, 8, 11, 12, 14, 16, 17, 18
tag recommender	5	1, 4, 5, 15, 18

Table 6.2: Feature usage of default tags (re-use of one or more tags from the previous item tagged), tag completion (completion of already known tags), and tag recommender (tags proposed by the recommender system below the tag line). TPS from the fast performers group are emphasized.

6 Evaluation

Feedback question	<i>p</i> -values	folder $\sigma \pm \text{SE}$	tagstore $\sigma \pm \text{SE}$
preference filing	$p < 0.03$	3.61 ± 0.33	4.61 ± 0.27
preference re-finding	$p < 0.001$	3.17 ± 0.38	5.06 ± 0.17
filing speed	$p > 0.13$	3.17 ± 0.33	3.94 ± 0.39
re-finding speed	$p < 0.01$	3.22 ± 0.42	4.83 ± 0.27
managing image files	$p < 0.001$	3.33 ± 0.35	3.44 ± 0.33
managing text files	$p < 0.006$	4.89 ± 0.21	4.67 ± 0.23

Table 6.3: FE1: Feedback questions showed a significant tendency of user preferences towards tagstore (Scale 0–6, higher values indicate more agreement with the statement; Wilcoxon test and confidence interval of 0.95). The TPs statistically significantly preferred the tagstore condition for all questions except for filing speed.

Interviews

In the interview directly after the last task, the TPs generally showed a positive reaction to tagstore. Some TPs mentioned that they would rather have used their own files instead of test files. Some TPs said that the folder hierarchy was too simple and the pause was too short so that they remembered where they had put the files.

The positive aspects of multi-classification of tagstore was mentioned several times.

Several TPs reported that they would like to see a re-tagging feature and that they wanted to see more tags in the recommender system. Multiple TPs mentioned the tag completion feature in a very positive way: although they did not use it for actual completion, they liked it because the drop-down box showed them important cues on tag names they had already been using.

The complete German transcript of the interviews is contained in De Vocht et al. (2012).

Feedback Questionnaire

The TPs filled out a feedback questionnaire (see Figure 6.1 and Figure 6.2). Results from those feedback questionnaires, as shown in Figure 6.3, draw

a positive picture of tagstore: all but one question showed a statistically significant positive result for the tagstore condition.

The question number thirteen of this questionnaire was: “All in all, which condition would you prefer?” Four of the TPs answered with the folder condition and 14 TPs answered with the tagstore condition. This is a 78 percent majority for the tagstore condition.

6.3.3 Discussion

The objective measures show that, in general, filing into folders performs faster (not significantly). Within the sub-group of fast performers, filing files with the tagstore condition was faster (not significantly). For filing, there is no clear winner.

Being in the fast performers group does not seem to be related to tagging experience: only three of the nine fast performers have used tagging before. The nine tagging-savvy people actually performed slower on average.

Re-finding performance was more or less the same for both conditions. Although re-finding in the tagstore condition required significantly less mouse clicks to get to the test files.

Figure 6.20 shows an interesting aspect: due to the higher number of tags compared to the number of (parent) folders per file, there are more associated terms within tagstore. This could be interpreted as an additional re-finding benefit for items stored in tagstore: items are associated with a richer set of metadata. This way, the user gets more associative cues within tagstore.

Positive aspects like additional re-find cues or higher user acceptance are not visible in the objective measures. Subjective feedback from the interview phase and from the feedback questionnaire was very positive for tagstore. Only one of the six directly-comparing questions did not show statistically significant positive results for the tagstore condition. No TP encountered difficulties while working with this new method. The short introduction video was enough to accomplish the tagstore tasks without any issues. No TP showed explicit negative feelings towards this new method.

6 Evaluation

On the contrary: TPs “defended” the tagstore implementation when they were affected by the umlaut bug.

Convenience features such as re-using the tags of the previous item or tag recommendation were used by almost all TPs. Although those features were not part of the introduction video, the users appreciated them. Even though only five TPs used the tag completion feature for completing tags, it was mentioned multiple times for being a great help on tagging. Tagging systems should provide similar convenience features to ease the process of tagging.

Summarizing the results, the tagstore implementation does not show any clear advantages or disadvantages when compared to traditional filing in folder hierarchies. However, with its enhanced possibilities for re-finding files (using significantly less mouse clicks) and its great user acceptance, tagstore is a clear benefit for users.

6.4 Formal Experiment 2

The previous section described the first formal experiment conducted with tagstore. It compared filing files with traditional methods and filing files using tagstore. The results of that experiment showed some positive aspects of tagstore. However, there were some issues with this experiment:

- The pause between filing and re-finding was too short.
- The number of test files was too small, resulting in very simple folder hierarchies.
- The tagstore implementation showed minor bugs.
- The TPs had to use artificial items and solve artificial tasks.

The last issue mentioned can only be addressed with field studies. The other issues could be addressed by a slightly changed experimental design. Therefore, a second formal experiment was conducted. It took place in April 2011 and is described in detail in Harzl et al. (2012). This section does not contain all results that were derived from the data. For the complete set of results regarding the feedback please refer to Harzl et al.

(2012). This thesis covers only the most important findings. All relevant data concerning this experiment is published on github⁴¹.

This experiment is very similar to the previous experiment, except for following differences:

- 27 test persons (TP) (instead of 18)
- 60 test files (instead of 30)
- two weeks pause between sessions (instead of a minimum of fifteen minutes)
- re-finding of ten files (instead of six)

6.4.1 Methodology

This study was also a repeated measures (within-groups) formal experiment. A set of 27 German-speaking TPS was split into two random groups. One pilot test was conducted before the experiment.

The methodology is the same as described in Section 6.3.1. The following sections describe only the changed parameters.

Procedure

In order to optimize the experimental design, the procedure changed slightly, and the background questionnaires were re-designed, as shown in Figure 6.21 and Figure 6.22.

We used sixty test files⁴² on general topics. Reviewing of the sixty test files was done in a separate room where TPS could use a standard Windows desktop computer. They were guided to this room and instructed by the person welcoming the TPS outside the experiment room. All tasks were printed on paper and given to the TPS. There was no time limit for the reviewing task and no other person was in the reviewing room besides the

41. <https://github.com/novoid/2011-04-tagstore-formal-experiment> – retrieved on 2012-08-26.

42. The set of test files consisted of 20 articles, 20 files consisting of graphical content, and 20 photographs.

6 Evaluation

Datum: 13.4.2011 Uhrzeit: 17:19 Testperson Nr.: T18

Hintergrundbefragung

Geschlecht: ☒ weiblich ☐ männlich

Alter: 48

Beruf: Sekretärin

Ausbildung

Abgeschlossene Ausbildung:

☐ Lehre ☒ Matura ☐ Studium ☐ Doktorat

Wenn Sie studieren oder studiert haben, beschreiben Sie bitte ihr Hauptstudiengebiet:

TECHN. PHYSIK

Umgang mit Computer

Wie lange benutzen Sie bereits einen Computer?

25 JAHRE

Wie viele Stunden pro Woche verwenden Sie einen Computer?

~~8~~ 40-50

Besitzen Sie einen eigenen Computer?

☒ Notebook ☐ Standrechner ☐ nein

Besitzen Sie Administratorrechte auf ihrem Computer?

☐ ja ☒ nein ☐ weiß ich nicht

Welches Betriebssystem verwenden Sie am meisten?

☐ Windows7 ☐ WindowsVista ☒ WindowsXP ☐ Mac OS X
☐ Linux Ubuntu ☐ andere: _____

Welchen Datei-Browser verwenden Sie?

☐ Windows Explorer ☐ Finder ☐ weiß nicht ☐ andere: SEA HONKEY

Welche Tätigkeiten führen Sie am häufigsten auf ihrem Computer aus?

☐ Internet surfen ☒ Email & Office ☐ Textverarbeitung ☐ Software Entwicklung ☐ Spielen & Musik
☐ Fotos/Videos bearbeiten, speichern, verwalten

Figure 6.21: An example background questionnaire from TP18, page one. The header information was filled out by the facilitator, the questionnaire was filled out by the test person.

6.4 Formal Experiment 2

Ist Ihnen der Begriff Tagging bekannt? ☒ ja ☐ nein
Wenn ja, was verstehen sie darunter? STRUKTURIERTES DATENARLEGEN

Verwenden Sie Tagging? ☐ ja ☒ nein
Wenn ja, wo verwenden Sie Tagging am häufigsten? _____

Ungefähr wie viele Ordner würden Sie als ihre wichtigsten Ordner erachten? 30

Wie viele Emails befinden sich für gewöhnlich in ihrer INBOX (=Posteingang)? ☐ <10 ☐ 10-50 ☐ 50-500 ☒ >500

Haben Sie ihre eigene hierarchische Struktur in der Sie Ihre Emails archivieren, oder lassen Sie alle Emails in ihrer INBOX? ☒ eigenes Archiv ☐ INBOX

Wenn Sie eine eigene hierarchische Struktur haben, wann ordnen Sie Ihre Emails in diese ein? ☐ sofort ☒ binnen 1 Woche ☐ >1 Woche

Erfahrung mit Usability Tests

Haben Sie schon an einer Usability Studie teilgenommen? ☐ ja ☒ nein

Wenn ja: ☐ als Testperson ☐ Mitglied des Testteams

Was war das für eine Studie? ☐ Thinking Aloud Test ☐ Formal Experiment
☐ ich weiss es nicht mehr

Vielen Dank!

Figure 6.22: An example background questionnaire from TP18, page two.

6 Evaluation

TP. This procedure removed time pressure from the TPs and should lead to less task distractions through file inspection.

After the first session, we introduced an additional interview phase, which collected insights from the filing tasks.

Finding and re-finding were split into two separate sessions with two weeks in between. This longer period of time should lead to less recall of filing decisions. In combination with the higher number of test files, this should result in re-finding processes that rely more on the re-finding method and less on memorized locations or tags.

We asked for ten test files to be re-found by the TPs. Differences in re-finding performance should be more dominant with more test files. There were five questions with more detailed file descriptions and five questions with vague descriptions of the items to be retrieved.

Filling out the feedback questionnaires was done in front of the test computer. The feedback questionnaires for filing tasks are shown in Figure 6.23 and Figure 6.24. After re-finding tasks, the feedback questionnaires of Figure 6.25, Figure 6.26, and Figure 6.27 were used. The new design used the semantic differential technique.

The (translated) posed questions were:

- Folders
 - Q1 How did you like using Windows Explorer hierarchy for filing items in general?
 - Q2 How did you like using Windows Explorer hierarchy for filing images?
 - Q3 How did you like using Windows Explorer hierarchy for filing text items?
- tagstore
 - Q4 How did you like using tagstore for filing items in general?
 - Q5 How did you like using tagstore for filing images?
 - Q6 How did you like using tagstore for filing text items?
- General
 - Q7 Would you use tagstore on your own computer?

6.4 Formal Experiment 2

Datum: 13.4.2011 Uhrzeit: 18:11 Test Nr.: 1 Testperson Nr.: T 18

Feedback Formular

Bewerten Sie bitte anhand folgender Aspekte Ihre Vorliebe für die eine oder die andere Dateimanagement-Variante.

Wie fanden Sie die übliche Ordner Variante allgemein bei der Dateiablage?

schnell	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	übersichtlich
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unterstützend

Wie fanden Sie die übliche Ordner Variante für das Ablegen der Bilder?

schnell	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	übersichtlich
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	unterstützend

Wie fanden Sie die übliche Ordner Variante für das Ablegen der Textdateien?

schnell	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	übersichtlich
einfach	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unterstützend

Wie fanden Sie tagstore allgemein bei der Dateiablage?

schnell	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	übersichtlich
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unterstützend

Figure 6.23: An example feedback form from TP18, filing, page one. The header information was filled out by the facilitator, the rest of the form was filled out by the test person.

Q8 Which condition would you prefer?

For the re-finding tasks, the same questions were posed with “filing” replaced with “re-finding”.

Likert scale questions were designed using a semantic differential. The attributes for the filing tasks were the following, using a seven-point scale:⁴³

⁴³. In this questionnaire, we changed positive and negative values in an alternating fashion to avoid habituation effects between answers.

6 Evaluation

Wie fanden Sie tagstore für das Ablegen der Bilder?

schnell	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	übersichtlich
einfach	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	unterstützend

Wie fanden Sie tagstore für das Ablegen der Textdateien?

schnell	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	übersichtlich
einfach	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	unterstützend

Könnten Sie sich vorstellen tagstore auf ihrem Computer einzusetzen?

Ja, auf jeden Fall ☒ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Nein, niemals

Weswegen? _____

Welche Variante würden Sie bevorzugen?

tagstore ☒ ☐ ☐ ☐ ☐ ☐ ☐ ☐ herkömmliche Ordner

Weswegen? MEHR BEGRIFFE, MEHR MÖGLICHKEITEN

ENTSCHEIDUNGSHILFE

Figure 6.24: An example feedback form from TP18, filing, page two.

fast	...	slow
good overview	...	chaotic
simple	...	complex
supportive	...	time-consuming

In the re-finding questionnaires, the attributes were more detailed:

fast	...	slow
chaotic	...	structured
simple	...	complex
time-saving	...	time-consuming
good overview	...	bad overview
intuitive	...	unintuitive
familiar	...	unfamiliar
positive	...	negative

6.4 Formal Experiment 2

Datum: 27.4.11 Uhrzeit: 15:25 Testperson: T18 Test Nr.: 2

Feedback Formular

tagstore

Bewerten Sie bitte anhand folgender Aspekte Ihre Vorlieben für **tagstore**.

1. Wie fanden Sie **tagstore Ordner** Variante beim Wiederfinden der Dateien?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

2. Wie fanden Sie **tagstore Ordner Variante** für das Wiederfinden der **Bilder**?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

3. Wie fanden Sie **tagstore Ordner Variante** für das Wiederfinden der **Textdateien**?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

Figure 6.25: An example feedback form from TP18, re-finding, page one. The header information was filled out by the facilitator, the rest of the form was filled out by the test persons.

6 Evaluation

Datum: 27.4.11

Uhrzeit: 15:32

Testperson: TP18

Test Nr.: 2

Feedback Formular

Übliche Ordner Variante - Windows Explorer

Bewerten Sie bitte anhand folgender Aspekte Ihre Vorlieben für die übliche Dateimanagement-Variante.

4. Wie fanden Sie die **übliche** Ordner Variante **allgemein** beim Wiederfinden der Dateien?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

5. Wie fanden Sie die **übliche** Ordner Variante für das Wiederfinden der **Bilder**?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

6. Wie fanden Sie die **übliche** Ordner Variante für das Wiederfinden der **Textdateien**?

	trifft zu	trifft eher zu	trifft gar nicht zu	weder noch	trifft gar nicht zu	trifft eher zu	trifft zu	
schnell	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	langsam
chaotisch	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strukturiert
einfach	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	kompliziert
zeitraubend	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	zeitsparend
übersichtlich	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	unübersichtlich
intuitiv	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	nicht intuitiv
vertraut	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unbekannt
positiv	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	negativ

Figure 6.26: An example feedback form from TP18, re-finding, page two.

6.4 Formal Experiment 2

7. Bitte geben Sie an, inwieweit die folgenden Aussagen auf Sie zutreffen.

	ja, auf jeden Fall	eher ja	vielleicht	eher nein	nein, auf keinen Fall
Können Sie sich vorstellen, tagstore auf ihrem Computer einzusehen?	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Weswegen? ich besuche per online News, das Zeit spart

8. Welche Variante würden Sie nun bevorzugen?

- ☐ ich bevorzuge tagstore
- ☒ ich bevorzuge eher tagstore
- ☐ weder noch
- ☐ ich bevorzuge eher die übliche Ordner Variante
- ☐ ich bevorzuge die übliche Ordner Variante

Weswegen? geres Konsept

Figure 6.27: An example feedback form from TP18, re-finding, page three.

Test Persons

The group of test persons for this experiment consisted of 27 persons (aged 32 on average, 33 percent female, 67 percent male) and was even more diverse than the previous group. Three persons could not make it to the second session. Therefore, the results and evaluation only contain data from 24 persons. Gender distribution in the chosen groups was the same. All but one of the TPs were students or possessed a university degree. Average computer usage per week was 46 hours with a very high standard deviation of 22.

Half of the test persons were using Microsoft Windows as their main system.⁴⁴ Tagging was used by 58 percent of the TPs. Table 6.4 gives an overview

⁴⁴ The rest of the TPs used GNU/Linux (29 percent), OS X (18 percent), and other systems

6 Evaluation

of the TPs. More details on the participants can be found in Harzl et al. (2012, Chapter 3.1).

Test Environment

Room The test room was a different one from the first experiment. The arrangement of furniture and hardware was the same.

Hardware The computer used for screening the items was a desktop computer: AMD Sempron 1.66 GHz, one Gigabytes of RAM, 1280×1024 TFT screen resolution, Microsoft Windows XP Home German (32Bit, SP3). All other tasks were accomplished on a notebook: lenovo SL500, intel Core 2 Duo 2.26 GHz, two Gigabytes of RAM, 1280×800 screen resolution on 15 inches, Microsoft Windows 7 Enterprise German. The observing person used a Sony Vaio notebook running Microsoft Windows 7 English.

Software The same Morae software version and setup was used in this experiment. The tagstore⁴⁵ software had a bugfix for the German umlaut bug. Once again, one tag line and no activated CV were used.

Training Training was done in the same neutral way as in the previous experiment.

Data Preparation The drive letter for the test files was I:. The language of the test computer was German. Figure 6.28 shows the tagstore filing situation for a TP.

(3 percent).

⁴⁵. The tagstore software used was Rev. 239 from 2011-04-01T14:46:06.

6.4 Formal Experiment 2

users	group	gender	uses tagging	OS	IT	filer/piler
TP01	1	m	×	other		filer
TP02	1	f		other		filer
TP03	1	m		other	×	piler
TP04	1	m		win		piler
TP05	1	m	×	other	×	piler
TP06	1	m	×	win		filer
TP07	1	f		win		filer
TP08	1	m	×	other	×	filer
TP09	1	f		other	×	piler
TP10	1	f		win		piler
TP12	2	m	×	win		filer
TP14	2	m	×	other	×	filer
TP15	2	m	×	win	×	piler
TP16	2	m	×	other	×	piler
TP17	2	m		win		filer
TP18	2	f		win		filer
TP19	2	m	×	win	×	piler
TP20	2	m	×	other		filer
TP21	2	m		win	×	filer
TP23	2	f		win		filer
TP24	2	f	×	other		piler
TP25	2	m	×	other	×	filer
TP26	2	f	×	win	×	piler
TP27	1	m	×	other	×	piler

Table 6.4: User groups and test users of the second formal experiment. [Source: Harzl et al. (2012)]

6 Evaluation

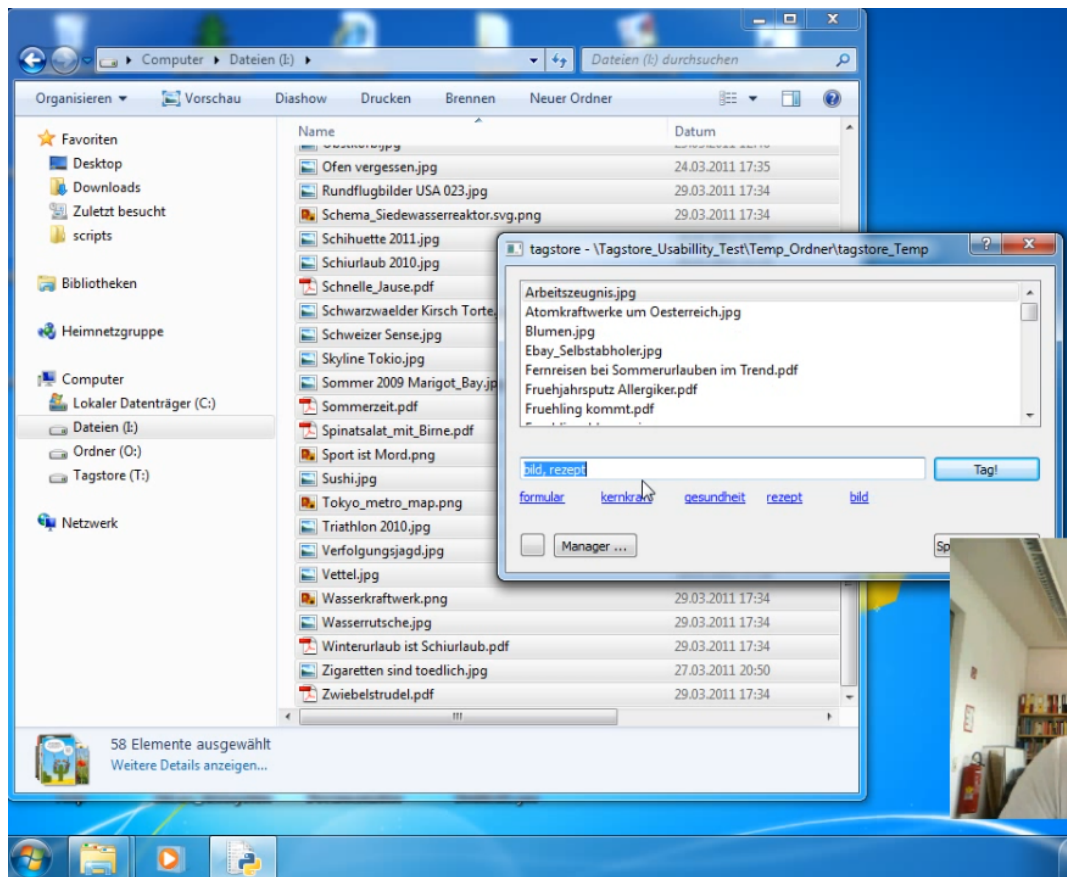


Figure 6.28: A screenshot from the Morae screencast: a TP filing in tagstore.

Data Processing

The **deterministic finite automata (DFA)** were simplified for this experiment because the transcription process for the first experiment was more detailed and took very long. This resulted in abandoning the explicit file management state of the folder condition (Figure 6.29) and feature logs of the tagstore condition (Figure 6.30).

The parser from the first experiment was able to parse the reduced set of log language as well. All logging files, the derived **csv** files and the parser

6.4 Formal Experiment 2

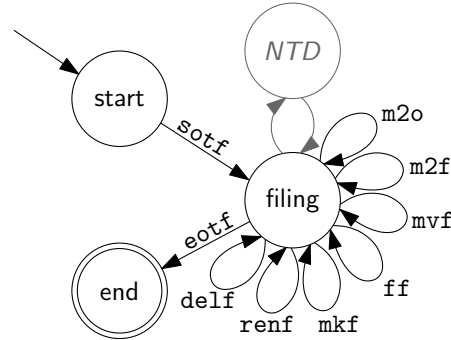


Figure 6.29: The deterministic finite automaton (DFA) for the filing task into a folder hierarchy for the second formal experiment. The file picking state (Figure 6.8) was abandoned to simplify the transcription process. This also resulted in less-detailed file management logs. The “NTD” state summarizes the non-task distractions which are shown in detail in Figure 6.10.

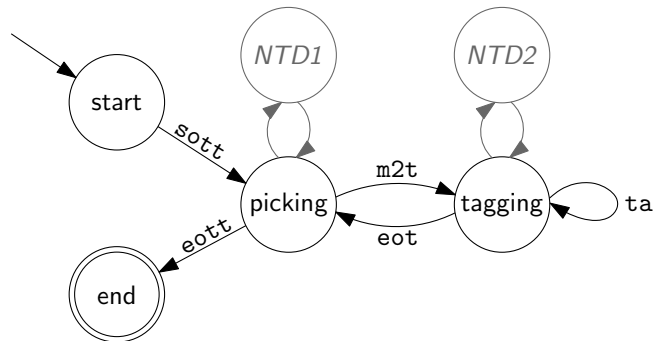


Figure 6.30: The deterministic finite automaton (DFA) for the tagging task into a folder hierarchy for the second formal experiment. In contrast to Figure 6.9 of the first experiment, feature usage and drive-changing transition rules were removed to simplify the transcription process. The “NTD” state summarizes the non-task distractions which are shown in detail in Figure 6.10.

6 Evaluation

	Folders	tagstore
Filing	678.67 ± 46.98	1044.79 ± 98.05
Re-finding	110.41 ± 9.99	141.69 ± 20.69

Table 6.5: Filing and re-finding times with mean values and standard errors. Filing in folders was significantly faster with $p < 0.01$ ($t(23) = -6.11$).

script were published on the github repository.⁴⁶

6.4.2 Results

Success Rate

For 240 re-find processes per condition, there was one unsuccessful re-finding process for the folder condition and four for the tagstore condition. This results in a success rate of 99.6 percent for the folder condition and 98.3 percent for the tagstore condition.

Task Performance

In Table 6.5, the task times for the folder condition are smaller for filing as well as for re-finding. For filing, there was a significant difference with $p < 0.01$ ($t(23) = -6.11$). For re-finding there was no significant difference ($p > 0.09$ with $t(23) = -1.73$).⁴⁷ Hence, folders were statistically significantly faster for filing.

The bar charts shown in Figure 6.31 and Figure 6.32 clearly show a number of sessions where the tagstore condition performed worse. Figure 6.33 and Figure 6.34 show the boxplot for filing and re-finding of both conditions.

The comparison of the average mouse clicks of each TP in both conditions did not reveal any significant difference ($p > 0.92$ with $t(24) = -0.10$).

⁴⁶. <https://github.com/novoid/2011-04-tagstore-formal-experiment> – retrieved on 2012-08-26.

⁴⁷. [git: results.org](https://git.results.org).

6.4 Formal Experiment 2

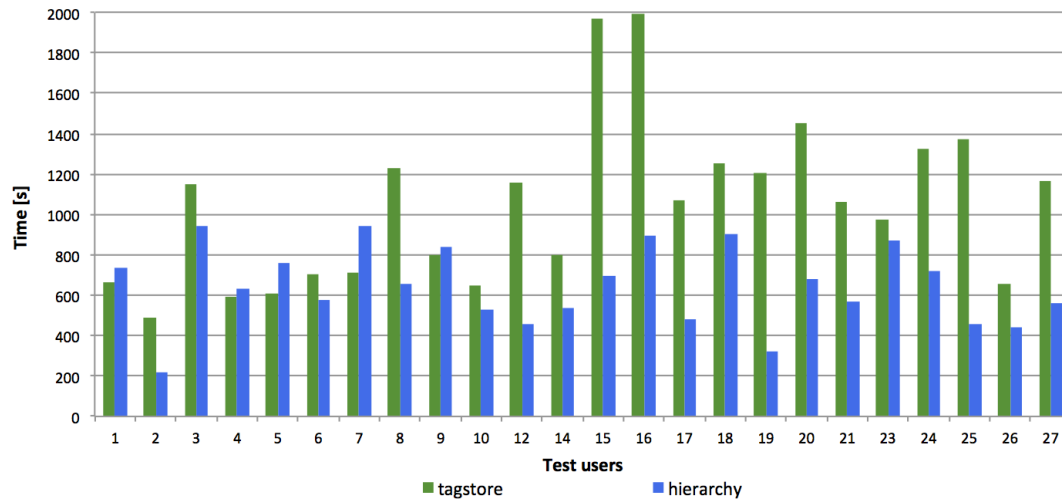


Figure 6.31: Bar chart for filing in both conditions.

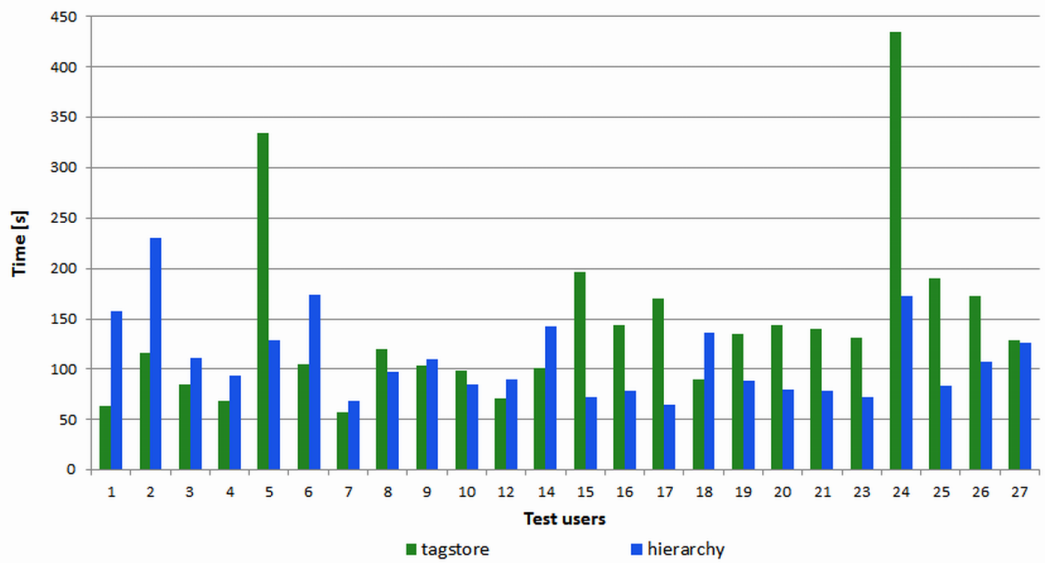


Figure 6.32: Bar Chart for re-finding in both conditions.

6 Evaluation

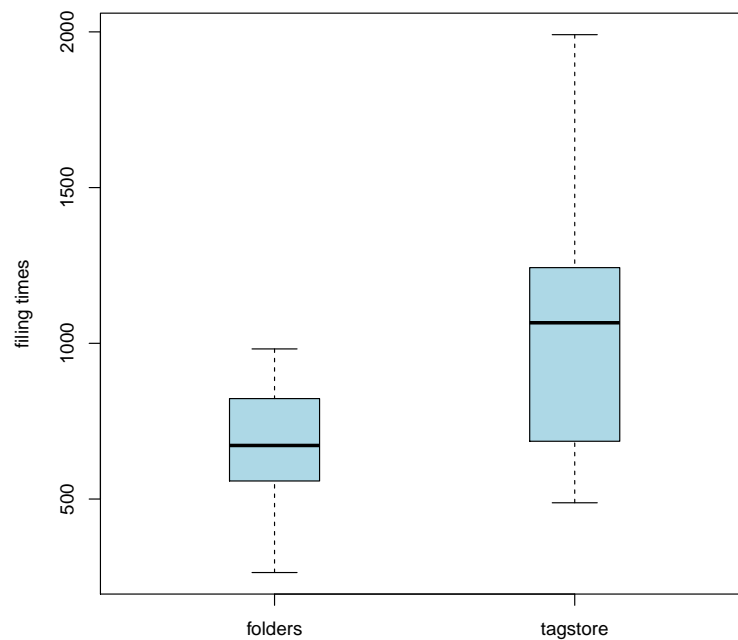


Figure 6.33: A boxplot of the filing tasks: the folder condition is significantly faster.

Besides evaluating general metrics as described above, we defined additional sets of persons and took a look into different comparisons.

Gender In order to find differences in the opinions which are related to gender, we examined the answers for each gender separately.

Platform All TPS were able to use Microsoft Windows and its Windows Explorer without any problem. But only half of them were using this system in their daily life. To find out, whether this is an issue, questionnaires were analyzed for Windows users and the rest.

IT background Almost all of our TPS (one exception which was abandoned in this metrics) were students or already had graduated from uni-

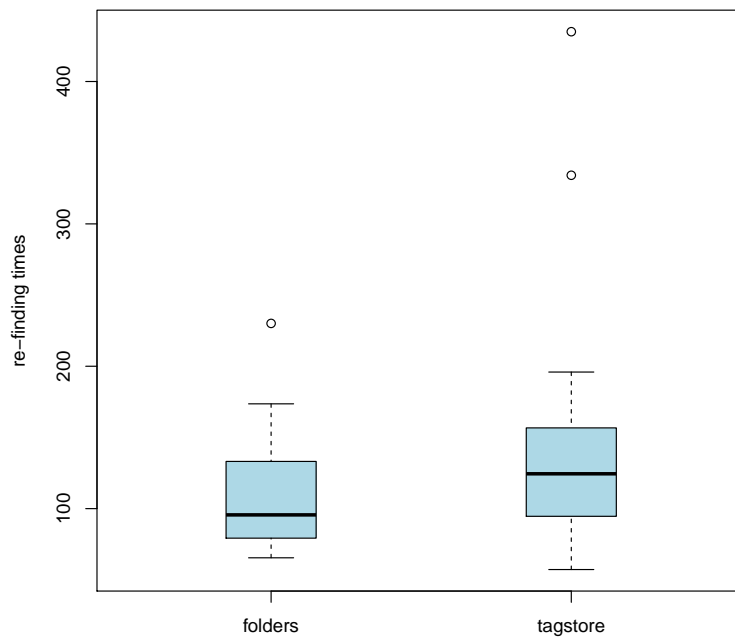


Figure 6.34: A boxplot of the re-finding tasks: no significant difference could be found for absolute refinding times of the TPs between both conditions.

versity. We divided them up in a set of IT-related background and a set of non-IT related studies.

Filer versus Piler A very common distinction regarding filing behavior is the simple metrics of **filer** and **piler**. In the background questionnaires we asked for the number of frequently used folders. As a very rough approximation, persons that used up to five folders were classified as pilers and all other were classified as filers.

Table 6.6 and Table 6.7 contain results from statistical analysis of the filing behavior in combination with the different sets of persons mentioned above. The only significant results occurred in the tagstore condition: females were faster than males, and the TPs who did not have tagging experience were faster than persons using tagging systems.

6 Evaluation

<i>Filing in folders</i>	groups		significance
gender	females	males	not significant
	721.75 ± 61.62	657.12 ± 39.04	$p > 0.23$ $t(15)=-1.24$
tagging experience	taggers	non-taggers	not significant
	639.00 ± 37.30	734.20 ± 57.31	$p > 0.67$ $t(13)=-0.44$
platform	Windows	other	not significant
	691.65 ± 53.11	602.88 ± 45.65	$p > 0.23$ $t(16)=1.23$
IT vs. other studies	IT studies	non-IT studies	not significant
	686.92 ± 46.39	740.18 ± 57.07	$p > 0.85$ $t(11)=-0.19$
filer vs. piler	filer	piler	not significant
	657.85 ± 49.11	703.27 ± 45.92	$p > 0.97$ $t(12)=-0.04$

Table 6.6: Filing in folder condition: overview of the results and statistical significance. Each group is listed with its mean value (time), standard error, and p -value. No statistically significant results were found.

<i>Filing in tagstore</i>	groups		significance
gender	females	males	<i>females faster</i>
	857.12 ± 72.81	1138.62 ± 103.00	$p < 0.004$ $t(15)=3.42$
tagging experience	taggers	non-taggers	<i>non-taggers faster</i>
	1166.00 ± 109.42	875.10 ± 63.63	$p < 0.006$ $t(13)=3.33$
platform	Windows	other	not significant
	1013.35 ± 82.28	1008.53 ± 101.44	$p > 0.96$ $t(16)=0.04$
IT vs. other studies	IT studies	non-IT studies	not significant
	1182.33 ± 109.44	1035.18 ± 127.77	$p > 0.18$ $t(11)=1.40$
filer vs. piler	filer	piler	not significant
	996.15 ± 72.84	1102.27 ± 123.89	$p > 0.66$ $t(12)=-0.45$

Table 6.7: Filing in tagstore condition: overview of the results and statistical significance. Each group is listed with its mean value (time), standard error, and p -value. Significant differences were found: females and persons with less tagging experience filed faster in tagstore.

6.4 Formal Experiment 2

<i>Re-finding in folders</i>	groups		significance
gender	females	males	not significant
	104.20 ± 7.86	122.85 ± 13.42	$p > 0.23$ $t(15)=-1.23$
tagging experience	taggers	non-taggers	not significant
	114.10 ± 8.70	105.26 ± 11.95	$p > 0.23$ $t(13)=1.27$
platform	Windows	other	not significant
	94.08 ± 7.05	118.97 ± 10.40	$p > 0.08$ $t(16)=-1.87$
IT vs. other studies	IT studies	non-IT studies	not significant
	105.89 ± 7.25	113.95 ± 12.97	$p > 0.49$ $t(11)=-0.71$
filer vs. piler	filer	piler	not significant
	113.59 ± 12.25	106.67 ± 6.93	$p > 0.44$ $t(12)=0.80$

Table 6.8: Re-finding in folder condition: overview of the results and statistical significance. Each group is listed with its mean value (time), standard error, and p -value. No statistically significant results were found.

For the re-finding tasks, there are four statistically significant differences which are listed in Table 6.8 and Table 6.9. Once again, significant differences could only be found in the tagstore condition: TPs who were not using tagging systems were faster, and TPs who did not study or graduate in an IT study were faster than IT-graduated persons.

Artifacts

The artifacts the TPs created had on average 7.6 characters per folder name and 7.5 characters per tag name (Figure 6.35). Comparing the number of (parent) folders for each file to the number of tags per file revealed 1.7 folders per file and 2.2 tags per file on average (Figure 6.36). For tagging, the TPs used many more different tags than folder names (Figure 6.37).

Interviews

Analysis of the transcripts of the interview phase (“How was it?”) revealed certain aspects which were mentioned multiple times.

6 Evaluation

<i>Re-finding in tagstore</i>	groups		significance	
gender	females	males	not significant	
	137.24 ± 16.12	150.59 ± 29.01	$p > 0.69$	$t(15)=-0.40$
tagging experience	taggers	non-taggers	<i>non-taggers faster</i>	
	167.13 ± 24.73	106.08 ± 8.29	$p < 0.03$	$t(13)=2.44$
platform	Windows	other	not significant	
	116.09 ± 10.10	160.63 ± 22.74	$p > 0.08$	$t(16)=-1.85$
IT vs. other studies	IT studies	non-IT studies	<i>other studies faster</i>	
	179.68 ± 25.22	101.37 ± 8.78	$p < 0.011$	$t(11)=3.10$
filer vs. piler	filer	piler	not significant	
	115.29 ± 9.73	172.88 ± 27.47	$p > 0.16$	$t(12)=-1.48$

Table 6.9: Re-finding in tagstore condition: overview of the results and statistical significance. Each group is listed with its mean value (time), standard error, and p -value. Significant differences were found: persons with less tagging experience and not having studied an IT-related study filed faster in tagstore.

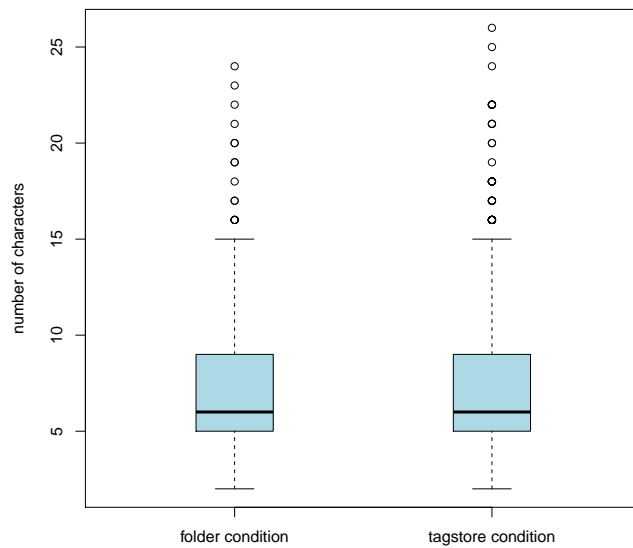


Figure 6.35: A comparison of the lengths of folder names and tag names showed that there is not much difference in the number of characters of folders and tags.

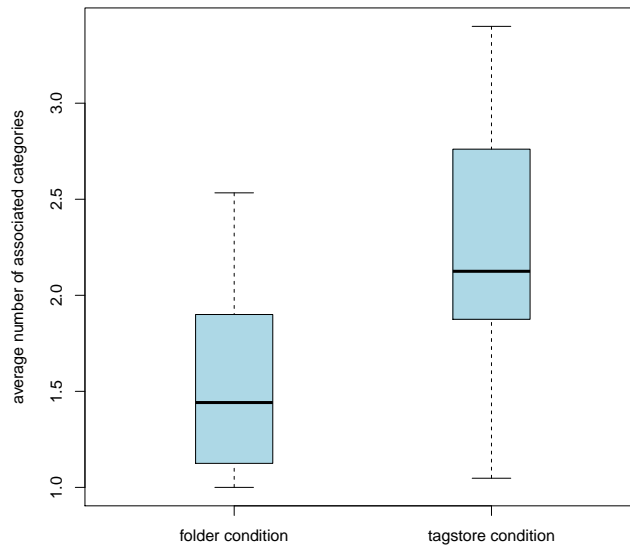


Figure 6.36: The number of associated (parent) folders of a file-path compared to the number of tags attached to files show that there are more tags associated to a file than folders.

Filing Folders Five TPS said that it was the method they are used to. Adjectives like “complex”, “exhausting”, and “taxing” were used multiple times. Some TPS remarked that it was not *their* files that had to be processed. Furthermore, they noted that some files would need multiple categories and not only one single folder.

Filing tagstore Six TPS noted that it would have been helpful to see more tag recommendations. Four mentioned explicitly, that they very much liked the feature of proposed tags. The auto-completing feature was mentioned multiple times in a very positive way. Adjectives like “pleasant” (3×), “not bad” (3×), “cool” (2×), “fast” (3×), “interesting”, “OK”, ... were used. Two TPS said that it was “tedious” to think about tags for every file.

Re-finding Folders Four TPS mentioned explicitly that this is the method that they are used to. Three TPS said that some files were hard to re-find.

6 Evaluation

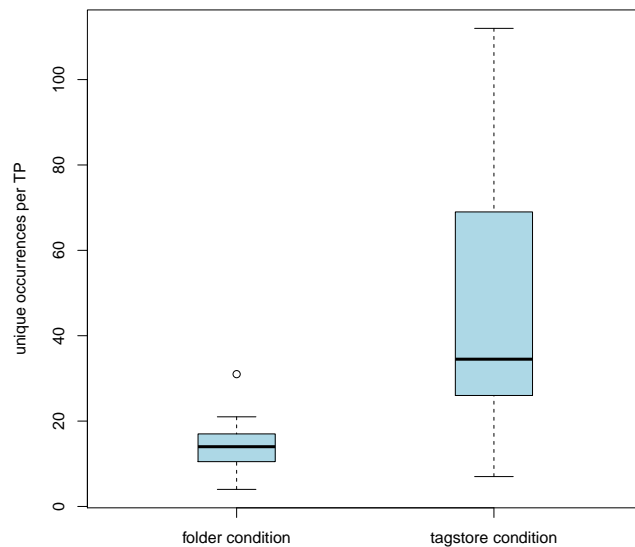


Figure 6.37: A boxplot comparing the number of (unique for each TP) folders with the list of tags (unique for each TP) of the TPs. There were many more different tags used than different folder names created.

Furthermore, three TPs mentioned that there were relatively few folders and that this was the reason the re-finding process was not too difficult. Two TPs indicated that the method was “not supportive” and “chaotic”. Two TPs stated that they preferred tagstore and two noted that they were able to re-find files quickly. It was also mentioned that for certain file types (photographs), the folder method is not suitable.

Re-finding tagstore Five TPs said that tagstore makes it fast to re-find files in an easy way. Four TPs said it was “simple”, “easy to use”, and “easy to re-find”. Words and phrases like “intuitive”, “supportive” (2×), “cool” (2×), “good”, “it works” (2×) were used. Two TPs noted that they had to remember which tags were used. One TP said that tagstore needs more time for filing, but makes re-finding less painful and faster through different associations. Several TPs mentioned that it is important to think about the tags to choose.

6.4 Formal Experiment 2

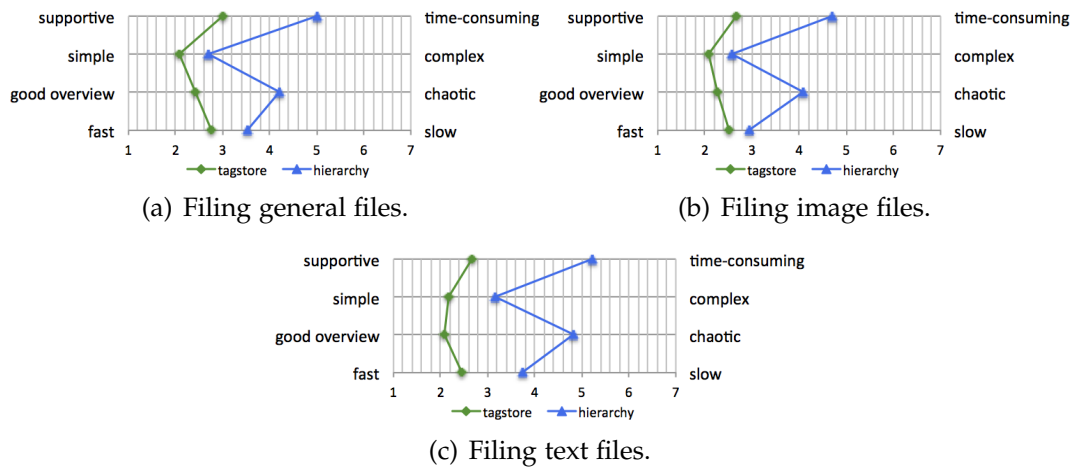


Figure 6.38: User feedback for filing different kind of files in both conditions.

Feedback Questionnaires

Since the subjective feedback of the TPs of the first formal experiment was very positive, the design and evaluation of the feedback questionnaires was done in an advanced fashion. The complete set of diagrams can be found in Harzl et al. (2012).

The results of the feedback questionnaires after the filing tasks show that users preferred the tagstore condition over the folder condition in the vast majority of cases. In all cases, the ratings for the tagstore condition (green) were generally more positive than the ratings for the folder condition (blue), with only very rare exceptions for individual data points.

Figure 6.38 shows feedback results for filing general files, image files, and text files. Feedback shows that the perceived differences had a maximum for filing text documents.

Male TPs gave very similar ratings for the three different file types (general, images, text). Female TPs produced more diverse feedback as shown in Figure 6.39.

Persons using Microsoft Windows as their main system gave more positive feedback for tagstore than TPs using other operating systems (Fig-

6 Evaluation

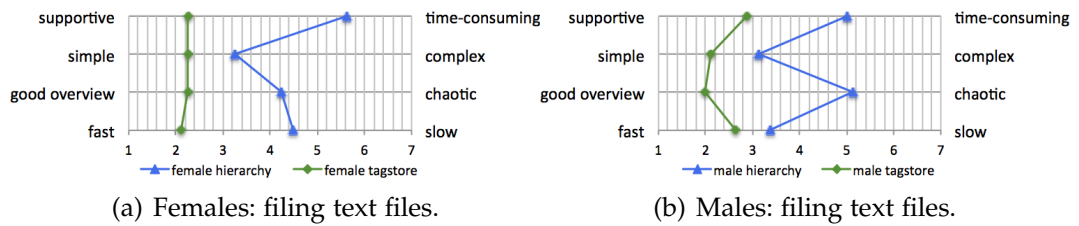


Figure 6.39: User feedback for filing text files: female TPs rating showed a different result with a more diverse picture.

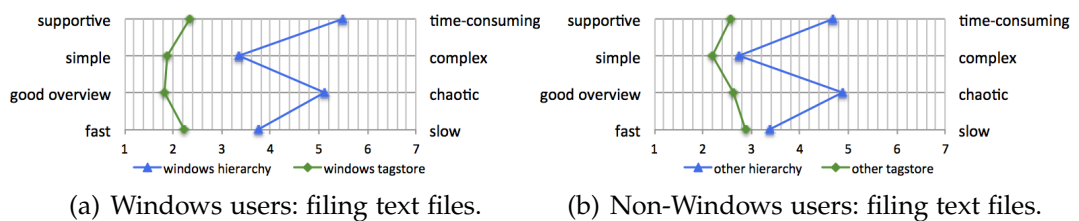


Figure 6.40: Users of Microsoft Windows rated the tagstore condition more positively than users of other operating systems.

ure 6.40).

If a TP is studying or had graduated from a study related to IT, the feedback for filing text files in folders was more positive than for TPs with no IT study (Figure 6.41). For image files, the non-IT persons rated the folder hierarchy better than the others.

User acceptance for tagstore after the filing tasks was very high: 78 percent of the TPs would prefer the tagstore condition over the folder condition (Figure 6.42) and 88 percent agreed to the question whether or not they would use tagstore on their own computer (Figure 6.43).

The results of the questionnaires after the re-finding tasks showed a similar tendency towards the tagstore condition. In general, TPs found the tagstore condition more intuitive, simple, time-saving, structured and faster than the hierarchy condition. Compared to the folder condition, tagstore was also better for providing a good overview. The only attribute where the folder condition showed better results in general was familiarity. Figure 6.44 shows the comparison between the feedback for the conditions

6.4 Formal Experiment 2

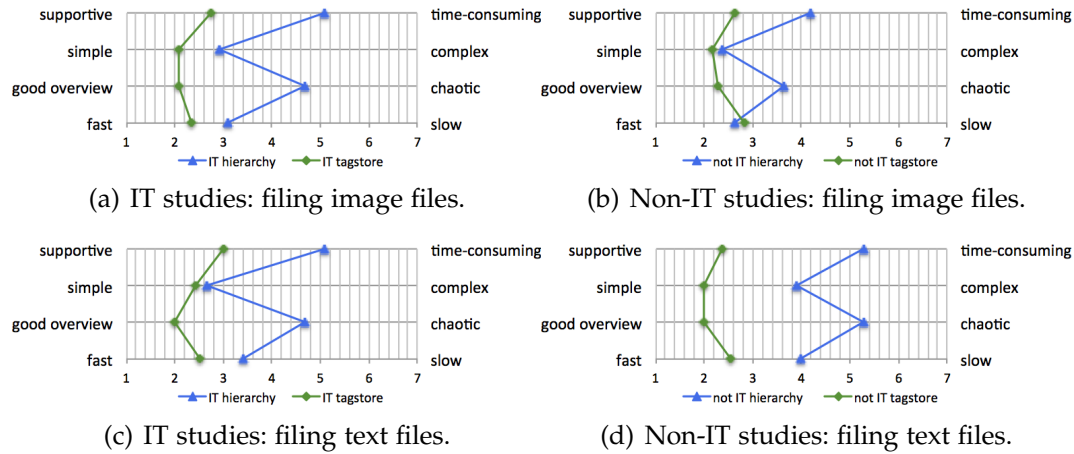


Figure 6.41: Diverse feedback between IT studies and non-IT studies for filing.

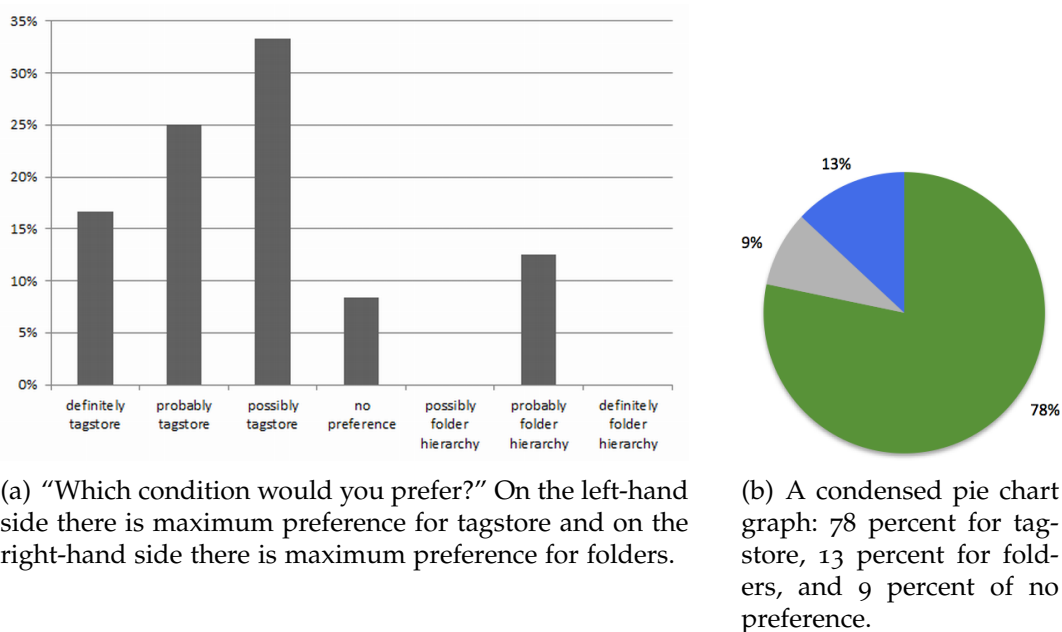


Figure 6.42: The majority of TRS would prefer the tagstore condition over the folder condition after the filing tasks.

6 Evaluation

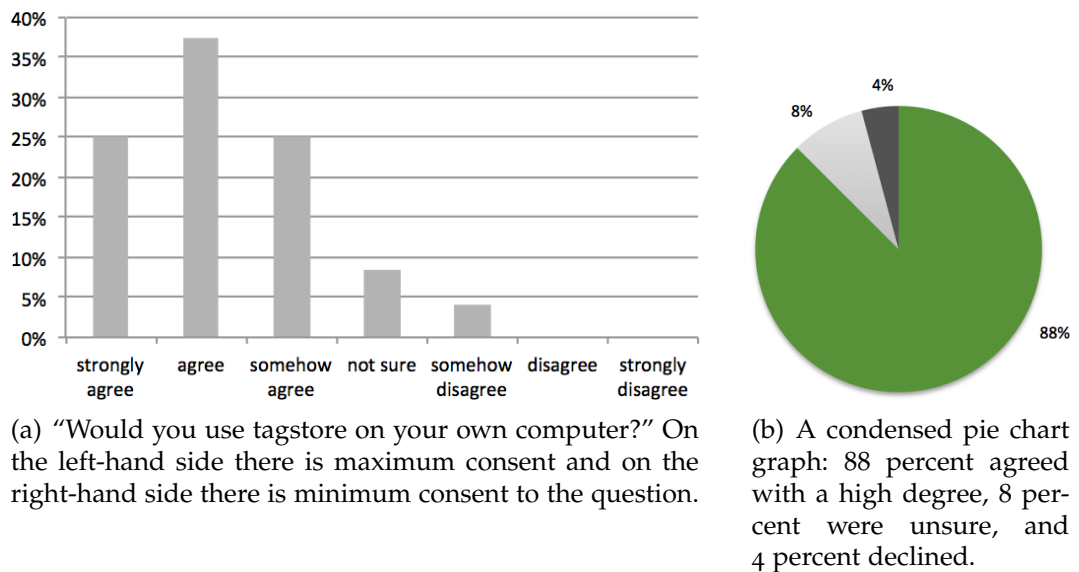


Figure 6.43: The majority of TPs would use tagstore on their own computer after the filing tasks.

after the re-finding tasks. On average, a TP found the tagstore condition to be faster than the folder condition. However, only ten of 24 TPs were faster for re-finding with the tagstore condition.

Analyzing the results from the questionnaires separated by gender (Figure 6.45) revealed that female users favored the tagstore condition to a greater extent than male users. For the filing tasks feedback, it was the other way round.

The difference between the two conditions is higher for TPs who do not have tagging experience. They also found that the tagstore condition was simpler for the re-finding tasks. Figure 6.46 visualizes the different ratings for taggers and non-taggers. The non-taggers also rated hierarchy for re-finding as much more time-consuming than the tagstore condition.

Users of operating systems other than Microsoft Windows rated the tagstore re-finding condition for general items as more familiar than the folder condition. Furthermore, they rated the folder condition as more structured than tagstore. The average numbers mentioned in this paragraph differed

6.4 Formal Experiment 2

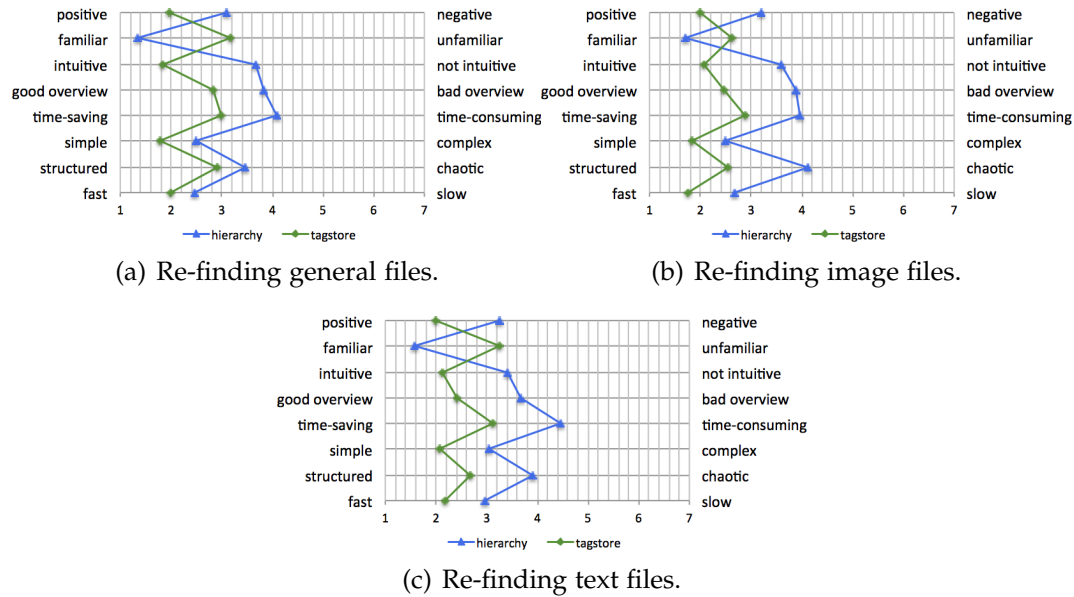


Figure 6.44: User feedback for re-finding different kinds of files in both conditions.

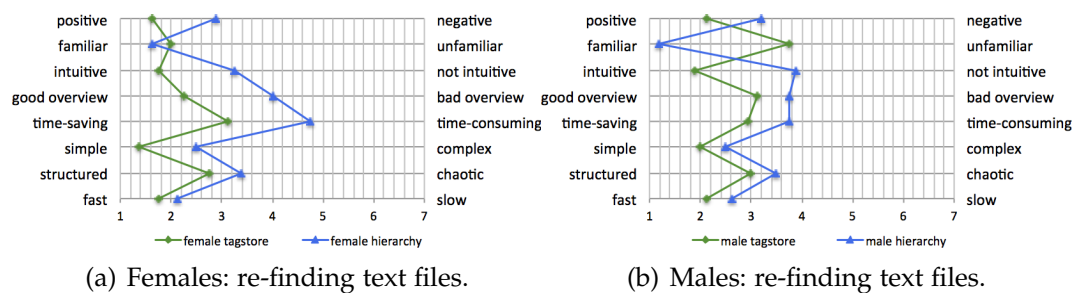


Figure 6.45: User feedback for re-finding text files: female TPs rated the tagstore condition as more positive than males.

6 Evaluation

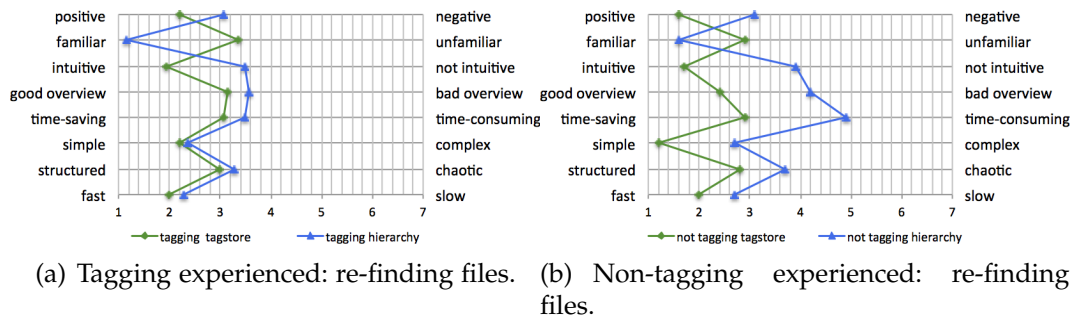


Figure 6.46: Tagging experienced TPUs rated tagstore better than non-taggers and found tagstore more simple than the folder condition for re-finding.

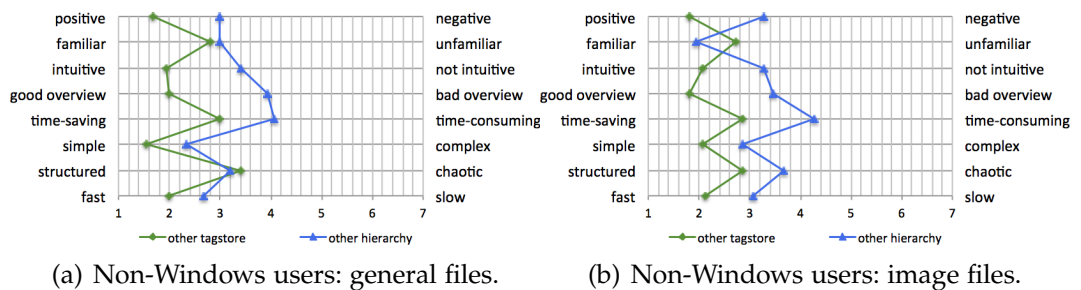


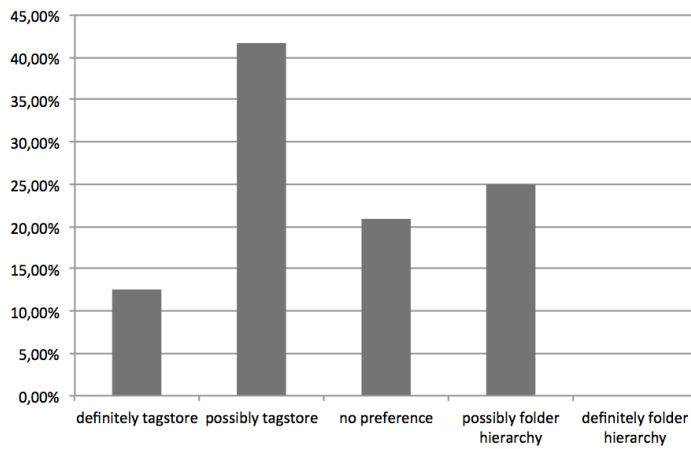
Figure 6.47: Left: The only case where the folder condition was slightly less familiar and more structured was for TPUs who do not use Microsoft Windows as their main system. Right: Re-finding specific file types such as images reversed this fact, showing the usual rating differences.

only by a small fraction, as shown in Figure 6.47(a). This was the only case where the folder condition was more unfamiliar and more structured than tagstore. Figure 6.47(b) shows the same set of TPUs rating re-finding of image files which did not result in this exceptional ratings.

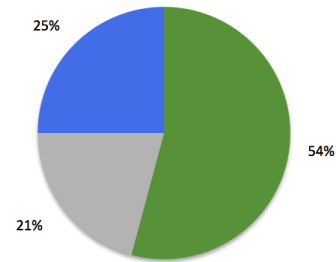
Some aspects of the rating results for TPUs who studies or graduated an IT study revealed a more positive rating for the tagstore condition than the group of academics of non-IT studies.

Figure 6.48 summarizes the answers for the question of which condition a TPU prefers after re-finding. More than half of the TPUs preferred the tagstore condition and only a fourth of the TPUs preferred the folder condition.

6.4 Formal Experiment 2

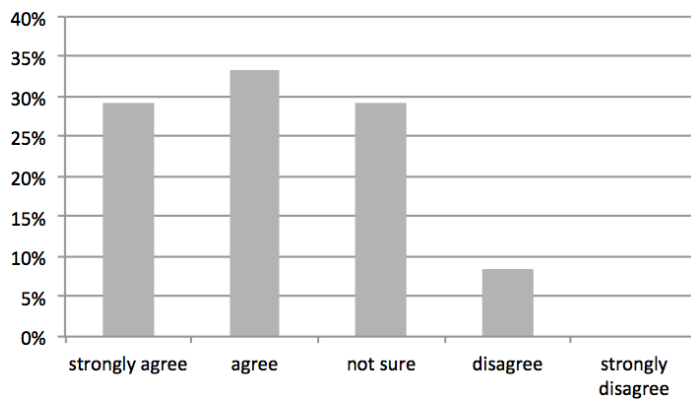


(a) “Which condition would you prefer?” On the left-hand side there is maximum preference for tagstore and on the right-hand side there is maximum preference for folders.

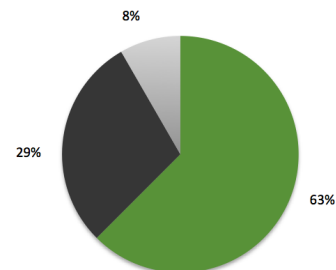


(b) A condensed pie chart graph: 54 percent for tagstore, 25 percent for folders, and 21 percent no preference.

Figure 6.48: The majority of TPs would prefer the tagstore condition over the folder condition after the re-finding tasks.



(a) “Would you use tagstore on your own computer?” On the left-hand side there is maximum consent and on the right-hand side there is minimum consent to the question.



(b) A condensed pie chart graph: 63 percent agreed with a high degree, 29 percent were unsure, and 8 percent declined.

Figure 6.49: The majority of TPs would use tagstore on their own computer after the re-finding tasks.

6 Evaluation

User acceptance was even higher for the question of whether or not the TPS would like to use tagstore on their own computer (Figure 6.49). Slightly less than two thirds would like to use tagstore and only eight percent would not like to use tagstore. The rest was unsure.

All feedback ratings showed that tagstore is the preferred condition of the TPS.

6.4.3 Discussion

Success rates for re-finding were above 98 percent in any case. Hence, it is not a very suitable measure to judge the effectiveness of both conditions.

Folders performed significantly better for filing tasks. Statistical analysis of the conditions showed that female TPS who were not using tagging systems themselves were performing best when filing in tagstore.

For re-finding, it was remarkable that once more, people not using tagging systems themselves were significantly faster using tagstore. This result contradicts the logical assumption that persons who use tagging themselves have an advantage in task times. The reason for this might be that the TPS using tagging systems want to think more carefully about the tags they use. However, this obviously did not help for task times in the re-finding phase, where non-tagging TPS outperformed the other TPS once more.

Furthermore, TPS who did not study or finish an IT study outperformed the TPS who studies or graduated from an IT study in re-finding. This is quite interesting, because one might think that IT-savvy persons have an advantage with new types of interfaces and methods. That result should be inspected further.

Artifacts showed that the metadata associated with items in tagstore was richer than in folders (Figure 6.36). This is an advantage for the re-finding process and shows that the tagging effort for TPS using tagstore does not seem to be that taxing.

The interviews showed a very positive and open-minded attitude of the TPS who favored tagstore. They associated much more positive attributes

to tagstore than to the folder condition. Subjective impression about time consumption and ease of use was much better for tagstore even though objective numbers contradicted from time to time.

Detailed analysis of the feedback questionnaires revealed interesting facts. First of all, tagstore was rated more positively than the folder condition in almost every case. Only the absolute difference to the ratings of the folder condition varied.

For filing, females who were using Microsoft Windows, not having studied IT and not using tagging systems before, rated tagstore in a more positive way. This has some similarity to the task time performance mentioned above.

After the re-finding tasks, TPs showed very positive ratings for tagstore as well. Only familiarity was rated higher for the folder condition. This seems obvious, since no TP had used tagstore before, in contrast to Microsoft Windows and its Windows Explorer.

Re-finding with the tagstore condition was rated best by females who did not use tagging systems, were working on a non-Windows operating system, and studies/graduated from an IT study.

The questions for the preferred condition of whether or not to use tagstore on one's own computer revealed a huge majority favoring tagstore. However, the numbers decreased from the first questionnaires after filing to the second ones after re-finding. This positive user acceptance shows that there is a desperate need for an alternative file management method like the one tagstore provides.

Probably the most interesting result is the different picture that arises from the objective numbers and the subjective feedback and ratings. On the one hand, task times reveal an advantage for the folder condition (significant only for filing). On the other hand, subjective measures show a clear tendency towards the tagstore condition. User acceptance is much higher for tagstore than for the traditional folders method, which the TPs were using for many years.

This poses the question whether or not [PIM](#) research should aim for efficiency (task times) or user acceptance and user satisfaction. These results

6 Evaluation

gave evidence that the both do not necessarily go hand in hand with each other.

And I'm thinking what a mess we're in
Hard to know where to begin

Virtual Insanity
Jamiroquai

7 Summary and Outlook

Several great [Personal Information Management \(PIM\)](#) research papers were presented in Chapter 2. Some of them provided ground-breaking visions, some gave insight from a psychological perspective, some presented novel tools. Many ideas resulted in clever prototypes and were evaluated in significant studies.

Chapter 3 discussed some key aspects and challenges, for which I find it very important to do further research. Changing parameters and environments require new approaches for [PIM](#). Search is a critical research topic for a large number of applications. However, navigation represents a crucial method as well. I presented several different factors which emphasize the need for better navigational methods. The strict hierarchical system of folders is a limiting factor to most people. It does not allow multi-classification and re-finding becomes a tedious task.

I developed the *TagTrees* method as a possible new approach to enhance navigational access for local items (files and folders) and to provide an alternative to the strict folder hierarchy. User-assigned tags are used to derive TagTrees, which are mapped onto the file system layer. Many different navigation paths lead to every single item, allowing for associative navigation. To access an item, no paths have to be remembered; intuitive cues are sufficient to re-find information.

7 Summary and Outlook

This TagTrees method was implemented in a research software framework called *tagstore*. It not only provides a strict implementation of the method, but also contains several other aspects which target end user convenience and acceptance. The tagging process should not be seen as a burden, but as a chance to express oneself. Maximum usability and features to support users in many ways was our goal. For researchers, tagstore includes a number of interesting features that qualify tagstore as being a test framework for studies on controlled vocabulary, multiple tag lines, recommender systems, and so forth.

Long-term users gave valuable feedback during the whole project, starting with the first prototype. Two formal experiments were conducted to obtain objective measures as well. The second experiment showed that, for filing tasks, the traditional folder structure performs statistically significantly faster than tagstore. The first experiment showed that users needed less mouse clicks for re-finding in tagstore. A large number of metrics gives insight to many interesting facts, such as tagging-savvy participants usually performing worse than participants who did not use tagging systems before. Although the objective numbers do not state a clear winner, the subjective feedback does. Almost all metrics we could think of favored the tagstore implementation over the traditional folder method. User acceptance was overwhelmingly in favor of tagstore.

The main contributions of this dissertation are a through summary of previous work in the topic of [PIM](#), the development of the TagTrees method, the tagstore implementation, and evaluations in multiple long-term and short-term studies.

Results show that navigational re-finding can be improved using an alternative method like TagTrees. Subjective user feedback was very explicit and positive in favor of tagstore and also showed that users like the notion of multi-classification and associative navigation. They organize information in a better way compared to the strict hierarchies.

A larger field study would be able to show implementation optimizations and more results related to long-term behavior of a more diverse set of participants.

Although the evaluation of the studies was very elaborate, there are so many possible metrics that we could not derive or probably even imagine them all. Many aspects concerning the test user artifacts can be part of further research work. We documented clearly how the artifacts were collected and published everything from the tagstore implementation and the raw data, to the scripts processing this data, up to the results and analyses. Other researchers are able to download this data and come up with new interpretations and different evaluation ideas.

With the free availability of the tagstore framework, researchers can use it to conduct experiments with aspects of [controlled vocabulary](#), multiple tag lines containing different types of tags, recommendation systems, expiry dates, and much more. The software can even be extended with additional features and configurations in an easy way.

Promising research should also be done by observing the change of tagging behavior over time. There are references to the fact that users need a certain amount of active tagging time to develop “their system” which they keep using for a longer period of time. I think that more research on [categorizer](#) and [describer](#) user types would lead to important results on tagging behavior.

Since tagstore is a *research* software, it has some practical limitations. To use tagstore over a longer period of time with a large number of items, further development should derive better ways for mapping tags to navigational paths. Other software tools use databases or special kinds of file systems. If this could be achieved without complicating the product, tagstore could be moved from a research software to a product.

Using folders to represent metadata such as tags, is a clever way to integrate a method like TagTrees into legacy software environments. However, this is just a workaround solution. Developing file systems which do not rely on a hierarchical internal representation, would be a first and important step. For a large number of things, hierarchical structures still work. Operating system files and end user software files might not need multi-classification at all. The part of the file system where users manage their files *does* need advanced file management features. TagTrees could be included into the file system level as well. From a technical perspective, *this* is the target layer TagTrees has to be implemented in.

7 Summary and Outlook

Moving features like multi-classification to the operating system and its underlying file system is a crucial step we have to make. Otherwise, end users need to spend even more effort for file management than they already do these days. A fully integrated TagTrees method would have additional possibilities for recommender systems which scan the content of items. Combined implicit and explicit semantic metadata would be able to provide an even better user experience.

TagTrees or tagstore is not the only method we should focus on. It is *one* possible part of a better PIM system out of many. It resembles a single (first) step towards a new direction which can be combined with other fresh ideas and best practices as well. With end users becoming more aware of PIM issues and proving that “different” can also be “better”, outdated patterns could be changed.

My long-term vision is a computer that does not need the concept of files or folders at all. Cutrell et al. (2006) mentioned that users do not want to remember file locations. The same paper mentioned that the file level is too monolithic and the haystack project (Section 2.2.7) used finer-grained information chunks as well. In fact, there have been systems that implemented this idea to a certain extent. For example, the *data soup* of Apple’s Newton Message Pad (Smith, 1994) resembled such an advanced paradigm. Currently, my most important data is managed and organized in Emacs Org-mode¹. Using a very simple text format as the common denominator wrongly seems to be a restriction to most people. Unlike this very minimalist approach, it offers the most elaborate and advanced feature-set I am aware of. With its Wiki-like structure, file borders do not limit my PIM anymore.

Many iterative processes and revolutionary attempts have to be made to achieve this vision. This is not an easy task. There is not only the problem of coming up with a solution in the first place. The migration from the current situation is probably just as difficult. Millions of software products and processes are bound to use files as the smallest common denominator. Accomplishing this huge task will require a broad consensus that there is a desperate *need* for change, great researchers, bright ideas, and many attempts to get the best PIM experience possible.

1. <http://orgmode.org> – retrieved on 2012-09-03.

Papers

This chapter summarizes papers, which are related to this thesis and were peer-reviewed by at least three reviewers. My contribution to the following papers includes everything except listed contributions from my co-authors.

The listed publications have already been published. Further results of this project will be part of future papers that are not yet published.

2009

Karl Voit, Keith Andrews, and Wolfgang Slany (Nov. 2009). “Why Personal Information Management (PIM) Technologies Are Not Widespread.” In: *PIM09 ASIS&T 2009 Workshop, Vancouver, BC, Canada*. Vancouver, BC, Canada: ACM, pp. 60–64. URL: <http://pimworkshop.org/2009/index.php?page=acceptedpapers>

Context The paper analyzed the situation of PIM in the field of file management regarding user acceptance and tool distribution. Eight requirements, which should be focused on, were described in detail.

Referring to chapters: Background

Co-authors Introduction, abstract, proofreading, structure, citation style, and reference check.

2011

Karl Voit, Keith Andrews, and Wolfgang Slany (Nov. 2011). “TagTree: Storing and Re-finding Files Using Tags.” In: *Proc. 7th Conference of the Austrian Computer Society Workgroup: Human-Computer Interaction (Usab 2011)*. Vol. 7058. LNCS. Graz, Austria: Springer, pp. 471–481. ISBN: 364-225-3636. DOI: [10.1007/978-3-642-25364-5_33](https://doi.org/10.1007/978-3-642-25364-5_33)

Context The tagstore implementation was introduced and described for the first time.

Referring to chapters: Implementation

Co-authors Citation style, reference check, proofreading, minor improvements and wording.

Karl Voit, Keith Andrews, Wolfgang Wintersteller, et al. (Mar. 2011). “TagTree: Exploring Tag-Based Navigational Structures.” In: *12th Internationalen Symposium der Informationswissenschaft (ISI)*. Ed. by Joachim Griesbaum, Thomas Mandl, and Christa Womser-Hacker. Vol. 58. Information und Wissen: global, sozial und frei? Hildesheim, Germany: Werner Hülsbusch, pp. 516–518. ISBN: 978-3-940317-91-9. URL: <http://www.vwh-verlag.de/vwh/?p=620>

Context The focus of this publication is the TagTrees method and its implications for information architecture and user experience.

Referring to chapters: TagTrees, Implementation

Co-authors Citation style, reference check, proofreading, tagstore overview, and some paragraphs from the Requirements sections.

2012

Karl Voit, Keith Andrews, and Wolfgang Slany (Feb. 2012a). “Creating a Comparative Environment for PIM Evaluation.” In: *PIM12 CSCW 2012 Workshop*. Seattle, WA, USA. URL: <http://pimworkshop.org/2012/papers>

Context This paper described tagstore as a flexible testing framework for conducting studies related to tagging. Providing a state-of-the-art tagging interface, which maps tag structures to the file system, allows for comparable study design.

Referring to chapters: TagTrees, Implementation, Evaluation

Co-authors Reference check, proofreading.

Karl Voit, Keith Andrews, and Wolfgang Slany (May 2012b). “Tagging Might Not Be Slower Than Filing in Folders.” In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems, Extended Abstracts (CHI2012)*. CHI '12. Austin, Texas, USA: ACM, pp. 2063–2068. ISBN: 978-1-4503-1016-1. DOI: [10.1145/2212776.2223753](https://doi.org/10.1145/2212776.2223753). URL: <http://dl.acm.org/citation.cfm?id=2212776.2223753>

Context Preliminary results from the first formal experiment were published in this paper. The very positive user feedback was one of the major results.

Referring to chapters: Evaluation

Co-authors Reference check, proofreading.

Bibliography

- Abrams, David, Ronald Baecker, and Mark H. Chignell (Apr. 1998). "Information Archiving with Bookmarks: Personal Web Space Construction and Organization." In: *Proceeding of the CHI '98 Conference on Human Factors in Computing Systems(CHI98)*. Ed. by Michael E. Atwood et al. Los Angeles, California, USA: ACM, pp. 41–48. ISBN: 0-201-30987-4 (cit. on p. 19).
- Allen, David (2001). *Getting Things Done – The Art of Stress-Free Productivity*. Penguin Books. URL: <http://www.gtdtimes.com/> (cit. on p. 72).
- Alvarado, Christine et al. (2003). *Surviving the information explosion: How people find their electronic information*. AI Memo AIM-2003-006. Department of Computer Science: MIT AI Laboratory. URL: <http://hdl.handle.net/1721.1/6713> (cit. on pp. 83, 234, 237).
- Anderwald, Johannes (2012). "tagstore: A Mobile Tagging Application with Synchronization." MA thesis. Graz, Austria: Graz University of Technology (cit. on p. 122).
- Baeza-Yates, Ricardo A. and Berthier Ribeiro-Neto (2011). *Modern Information Retrieval*. 2nd. Harlow, England: Pearson Education Ltd. ISBN: 978-0-321-41691-9 (cit. on pp. 68, 235, 236).
- Bakshi, Karun and David R. Karger (2005). "End-user Application Development for the Semantic Web." In: *Proceedings of the 1st Workshop on the Semantic Desktop: Next Generation Personal Information Management and Collaboration Infrastructure, located at the International Semantic Web Conference*. URL: <http://people.csail.mit.edu/kbakshi/SemDesk%20-%20Final.pdf> (cit. on pp. 29, 31).
- Bälter, Olle (Aug. 1997). "Strategies for Organising Email." In: *BCS HCI*. Ed. by Harold W. Thimbleby, Brid O'Conaill, and Peter Thomas. Springer, pp. 21–38. ISBN: 3-540-76172-1 (cit. on p. 19).

Bibliography

- Barreau, Deborah (June 1995). "Context as a Factor in Personal Information Management Systems." In: *Journal of the American Society for Information Science* 46.5, pp. 327–339. ISSN: 0002-8231. DOI: [10.1002/\(SICI\)1097-4571\(199506\)46:5<327::AID-ASI4>3.0.CO;2-C](https://doi.org/10.1002/(SICI)1097-4571(199506)46:5<327::AID-ASI4>3.0.CO;2-C) (cit. on p. 82).
- Barreau, Deborah (Jan. 2008). "The Persistence of Behavior and Form in the Organization of Personal Information." In: *Journal of the American Society for Information Science and Technology* 59.2, pp. 307–317. ISSN: 1532-2882. DOI: [10.1002/asi.20752](https://doi.org/10.1002/asi.20752) (cit. on p. 83).
- Barreau, Deborah and Bonnie A. Nardi (July 1995). "Finding and Reminding: File Organization from the Desktop." In: *SIGCHI Bulletin* 27.3, pp. 39–43. ISSN: 0736-6906. DOI: [10.1145/221296.221307](https://doi.org/10.1145/221296.221307). URL: <http://www.sigchi.org/bulletin/1995.3/barreau.html> (cit. on pp. 19, 64, 81).
- Bell, Gordon and Jim Gemmell (Sept. 2009). *Total Recall*. 1st. Dutton (cit. on pp. 42, 43, 45).
- Bergman, Ofer et al. (Sept. 2008). "Improved Search Engines and Navigation Preference in Personal Information Management." In: *Transactions on Information Systems* 26.4, pp. 1–24. ISSN: 1046-8188. DOI: [10.1145/1402256.1402259](https://doi.org/10.1145/1402256.1402259) (cit. on p. 84).
- Binder, Gerulf (2012). *Marktübersicht von Tagging-Werkzeugen und Vergleich mit tagstore*. Tech. rep. Graz, Austria: Graz University of Technology (cit. on p. 139).
- Binder, Kurt (1986). "Monte Carlo Methods in Statistical Physics." In: 2nd. Vol. 7. Topics in Current Physics. Springer Verlag, p. 411. ISBN: 978-0387-165-141 (cit. on p. 90).
- Bloehdorn, Stephan et al. (Sept. 2006). "TagFS – Tag Semantics for Hierarchical File Systems." In: *Proc. 6th International Conference on Knowledge Management (I-KNOW 06)*. Graz, Austria, pp. 304–312. URL: http://triple-i.tugraz.at/blog/wp-content/uploads/2008/11/37_tagfs.pdf (cit. on pp. 47, 96).
- Boardman, Richard (July 2001). "Category Overlap between Hierarchies in User Workspace." In: *Proceedings of IFIP INTERACT'01: Human-Computer Interaction*. Tokyo, Japan, pp. 759–760 (cit. on p. 32).
- Boardman, Richard and M. Angela Sasse (Mar. 2001). "Multiple Hierarchies in User Workspace." In: *Proc. 19th SIGCHI Conference on Human Factors in Computing Systems (CHI 2001) Extended Abstracts*. Seattle, Washington, USA: ACM, pp. 403–404. DOI: [10.1145/634067.634304](https://doi.org/10.1145/634067.634304). URL:

- <http://www.iis.ee.ic.ac.uk/~rick/research/pubs/workspace-chi2001.pdf> (cit. on p. 32).
- Boardman, Richard and M. Angela Sasse (Apr. 2004). ““Stuff Goes into the Computer and Doesn’t Come Out”: A Cross-Tool Study of Personal Information Management.” In: *Proc. 22nd SIGCHI Conference on Human Factors in Computing Systems (CHI 2004)*. Vienna, Austria: ACM, pp. 583–590. DOI: [10.1145/985692.985766](https://doi.org/10.1145/985692.985766). URL: <http://www.iis.ee.ic.ac.uk/~rick/research/pubs/boardman-chi04.pdf> (cit. on pp. 9, 17, 19, 20, 32, 33, 64, 71, 80, 83, 89, 101).
- Boardman, Richard, M. Angela Sasse, and Bob Spence (Nov. 2002). “Life Beyond the Mailbox: A Cross-Tool Perspective on Personal Information Management.” In: *Proc. CSCW 2002 Workshop on Redesigning Email for the 21st Century*. New Orleans, Louisiana, USA: ACM. URL: <http://www.iis.ee.ic.ac.uk/~rick/research/pubs/email-cscw2002.pdf> (cit. on p. 32).
- Bush, Vannevar (1945). “As We May Think.” In: *Atlantic Monthly* 176, pp. 101–108. URL: <http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/3881/> (cit. on pp. 10, 101).
- Chau, Duen Horng, Brad Myers, and Andrew Faulring (Apr. 2008a). “Feldspar: A System for Finding Information by Association.” In: *Proceedings of the Personal Information Management Workshop at the CHI 2008 (PIM2008)*. Ed. by Jaime Teevan and William Jones. Florence, Italy: ACM. URL: <http://www.cs.cmu.edu/~dchau/feldspar/feldspar-pim08.pdf> (cit. on pp. 49, 50).
- Chau, Duen Horng, Brad Myers, and Andrew Faulring (Apr. 2008b). “What to do When Search Fails: Finding Information by Association.” In: *Proc. 26th SIGCHI Conference on Human Factors in Computing Systems (CHI 2008)*. Florence, Italy: ACM, pp. 999–1008. DOI: [10.1145/1357054.1357208](https://doi.org/10.1145/1357054.1357208). URL: <http://www.cs.cmu.edu/~dchau/feldspar/feldspar-chi08.pdf> (cit. on pp. 49, 50, 61, 101, 237).
- Cho, Hyeyoung, Sungho Kim, and Sik Lee (Dec. 2009). “Analysis of Long-Term File System Activities on Cluster Systems.” In: *World Academy of Science, Engineering and Technology* 36, pp. 159–164. URL: <http://www.waset.org/journals/waset/v60/v60-28.pdf> (cit. on pp. 64, 65, 89, 104, 105).
- Chui, Michael et al. (July 2012). *The Social Economy: Unlocking Value and Productivity Through Social Technologies*. Tech. rep. McKinsey Global In-

Bibliography

- stitute. URL: http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/the_social_economy (cit. on p. 68).
- Cockton, Gilbert and Panu Korhonen, eds. (Apr. 2003). *Proceedings of the 2003 Conference on Human Factors in Computing Systems, CHI2003*. Ft. Lauderdale, Florida, USA. ISBN: 1-58113-630-7.
- Collins, Anthony (June 2011). "New dimensions of file access at tabletops: associative and hierarchical; private and shared; individual and collaborative." PhD thesis. Sydney, Australia: School of Information Technologies, University of Sydney. URL: <http://sydney.edu.au/engineering/it/~anthony/publications/AnthonyCollins-PhDThesis.pdf> (cit. on p. 101).
- Corbató, F. J. and V. A. Vyssotsky (1965). "Introduction and Overview of the Multics System." In: *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*. AFIPS '65 (Fall, part I). Las Vegas, Nevada: ACM, pp. 185–196. DOI: [10.1145/1463891.1463912](https://doi.org/10.1145/1463891.1463912). URL: <http://www.multicians.org/fjcc1.html> (cit. on p. 3).
- Cutrell, Edward et al. (Apr. 2006). "Fast, Flexible Filtering with Phlat." In: *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI 2006)*. Montréal, Québec, Canada: ACM, pp. 261–270. DOI: [10.1145/1124772.1124812](https://doi.org/10.1145/1124772.1124812). URL: <http://research.microsoft.com/en-us/um/people/cutrell/chi2006%20proceedings--phlat.pdf> (cit. on pp. 37–40, 89, 99, 208).
- Delescluse, Matthieu et al. (Aug. 2011). "Making neurophysiological data analysis reproducible. Why and how?" In: *Journal of Physiology Paris* (cit. on p. 53).
- De Vocht, Laurens et al. (Feb. 2012). *Formal Experiment Report: Tagging files vs. placing files in a hierarchy*. Tech. rep. Graz, Austria: Graz University of Technology. URL: https://github.com/novoid/2011-01-tagstore-formal-experiment/blob/master/analysis_and_derived_data/Results_Report.pdf (cit. on pp. 145, 170).
- Dominik, Carsten (2010). *The Org-Mode 7 Reference Manual: Organize Your Life with GNU Emacs*. with contributions by David O'Toole, Bastien Guerry, Philip Rooke, Dan Davison, Eric Schulte, and Thomas Dye. UK: Network Theory (cit. on p. 53).
- Dourish, Paul (2003). "The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents." In: *Computer Supported Cooperative Work (CSCW)* 12.4, pp. 465–490 (cit. on p. 28).

- Dourish, Paul, W. Keith Edwards, Anthony Lamarca, et al. (Apr. 2000). "Extending Document Management Systems with User-Specific Active Properties." In: *Transactions on Information Systems* 18.2, pp. 140–170. ISSN: 1046-8188. DOI: [10.1145/348751.348758](https://doi.org/10.1145/348751.348758). URL: <http://www.dourish.com/publications/2000/tois-placeless.pdf> (cit. on p. 28).
- Dourish, Paul et al. (June 1999a). "Presto: An Experimental Architecture for Fluid Interactive Document Spaces." In: *Transactions on Information Systems* 6.2, pp. 133–161. ISSN: 1073-0516. DOI: [10.1145/319091.319099](https://doi.org/10.1145/319091.319099). URL: <http://www.dourish.com/publications/1999/tochi-presto.pdf> (cit. on p. 28).
- Dourish, Paul et al. (Nov. 1999b). "Using Properties for Uniform Interaction in the Presto Document System." In: *Proc. 12th Annual ACM Symposium on User Interface Software and Technology (UIST'99)*. Asheville, North Carolina, USA: ACM, pp. 55–64. DOI: [10.1145/320719.322583](https://doi.org/10.1145/320719.322583). URL: <http://www2.parc.com/csl/projects/placeless/papers/uist99-presto.pdf> (cit. on pp. 28, 29).
- Dumais, Susan et al. (2003). "Stuff I've Seen: a System for Personal Information Retrieval and Re-use." In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '03. Toronto, Canada: ACM, pp. 72–79. ISBN: 1-58113-646-3. DOI: [10.1145/860435.860451](https://doi.org/10.1145/860435.860451) (cit. on pp. 33–36, 38–40, 80, 89).
- Dumas, Joseph S. and Janice C. Redish (1999). *A Practical Guide to Usability Testing*. 2nd. Lives of Great Explorers Series. Intellect Limited. ISBN: 9781841500201 (cit. on p. 142).
- Elsweiler, David (Nov. 2007). "Supporting Human Memory in Personal Information Management." PhD thesis. Department of Computer and Information Sciences, University of Strathclyde. URL: http://www.cis.strath.ac.uk/cis/research/publications/papers/strath_cis_publication_2328.pdf (cit. on p. 101).
- Engelbart, Douglas C. and William K. English (1968). "A Research Center for Augmenting Human Intellect." In: *Proceedings of the December 9–11, 1968, fall joint computer conference, part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, pp. 395–410. DOI: [10.1145/1476589.1476645](https://doi.org/10.1145/1476589.1476645). URL: <http://sloan.stanford.edu/MouseSite/> (cit. on pp. 22, 23).
- Faubel, Sebastian and Christian Kuschel (Oct. 2008). "Towards Semantic File System Interfaces." In: *International Semantic Web Conference (Posters*

Bibliography

- & Demos). Ed. by Amit P. Sheth et al. Vol. 5318. Lecture Notes in Computer Science. Karlsruhe, Germany: Springer. ISBN: 978-3-540-88563-4. URL: http://ceur-ws.org/Vol-401/iswc2008pd_submission_8.pdf (cit. on p. 87).
- Feldman, Susan and Chris Sherman (July 2001). *The High Cost of not Finding Information*. Tech. rep. IDC. URL: <http://www.idc.com> (cit. on p. 68).
- Fertig, Scott, Eric Thomas Freeman, and David Gelernter (Jan. 1996). "Finding and Reminding. Reconsidered." In: *SIGCHI Bulletin* 28.1, pp. 66–69. ISSN: 0736-6906. DOI: [10.1145/249170.249187](https://doi.org/10.1145/249170.249187) (cit. on p. 81).
- Freeman, Eric Thomas (May 1997). "The Lifestreams Software Architecture." PhD thesis. Yale University Department of Computer Science. URL: <http://www.cs.yale.edu/homes/freeman/dissertation/etf.pdf> (cit. on pp. 23, 25).
- Freeman, Eric Thomas and Scott Fertig (1995). "Lifestreams: Organizing your Electronic Life." In: *AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval*. Association for the Advancement of Artificial Intelligence, pp. 38–44. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.6769&rank=4> (cit. on pp. 23, 24).
- Freeman, Eric Thomas and David Gelernter (1996). "Lifestreams: A Storage Model for Personal Data." In: *ACM SIGMOD Bulletin* 25, pp. 80–86 (cit. on pp. 23, 234).
- Furnas, George W., Caterina Fake, et al. (Apr. 2006). "Why Do Tagging Systems Work?" In: *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI 2006) Extended Abstracts*. Montréal, Québec, Canada: ACM, pp. 36–39. DOI: [10.1145/1125451.1125462](https://doi.org/10.1145/1125451.1125462) (cit. on p. 89).
- Furnas, George W., Thomas K. Landauer, et al. (Nov. 1987). "The Vocabulary Problem in Human-System Communication." In: *Communications of the ACM* 30.11. Ed. by Henry Ledgard, pp. 964–971. DOI: [10.1145/32206.32212](https://doi.org/10.1145/32206.32212) (cit. on p. 84).
- Gemmell, Jim, Gordon Bell, and Roger Lueder (Jan. 2006). "MyLifeBits: A Personal Database for Everything." In: *Communications of the ACM* 49.1, pp. 88–95. ISSN: 0001-0782. DOI: [10.1145/1107458.1107460](https://doi.org/10.1145/1107458.1107460). URL: <http://research.microsoft.com/pubs/64157/tr-2006-23.pdf> (cit. on pp. 42–44, 64).
- Gemmell, Jim, Lyndsay Williams, et al. (2004). "Passive capture and ensuing issues for a personal lifetime store." In: *Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences*.

- CARPE'04. New York, New York, USA: ACM, pp. 48–55. ISBN: 1-58113-932-2. DOI: [10.1145/1026653.1026660](https://doi.org/10.1145/1026653.1026660) (cit. on p. 44).
- Gibson, Timothy J., Ethan L. Miller, and Darrell D. E. Long (Dec. 1998). "Long-term File Activity and Inter-Reference Patterns." In: *Int. CMG Conference*. Anaheim, California, USA: Computer Measurement Group, pp. 976–987. DOI: [10.1.1.34.2476](https://doi.org/10.1.1.34.2476). URL: <http://new.cmg.org/proceedings/1998/8100.pdf> (cit. on pp. 35, 66, 89, 104).
- Gifford, David K. et al. (Oct. 1991). "Semantic File Systems." In: *Proc. 13th ACM Symposium on Operating Systems Principles (SOSP 1991)*. Pacific Grove, California, USA: ACM, pp. 16–25. DOI: [10.1145/121132.121138](https://doi.org/10.1145/121132.121138). URL: <http://cgs.csail.mit.edu/history/publications/Papers/sfs.ps> (cit. on pp. 45, 46, 101).
- Golovchinsky, Gene and Abdigani Diriye (Oct. 2011). "Session-based search with Querium." In: *The Fifth Workshop on Human-Computer Interaction and Information Retrieval (HCIR)*. Mountain View, California, USA (cit. on p. 61).
- Gonçalves, Daniel J. and Joaquim A. Jorge (June 2003). "An Empirical Study of Personal Document Spaces." In: *Proc. 10th International Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS 2003)*. Vol. 2844. LNCS. Funchal, Madeira Island, Portugal: Springer, pp. 46–60. DOI: [10.1007/b13960](https://doi.org/10.1007/b13960). URL: <http://virtual.inesc.pt/dsvis03/papers/05.pdf> (cit. on p. 79).
- Haraty, Mona et al. (2012). "Individual Differences in Personal Task Management: a Field Study in an Academic Setting." In: *Proceedings of the 2012 Graphics Interface Conference*. GI '12. Toronto, Ontario, Canada: Canadian Information Processing Society, pp. 35–44. ISBN: 978-1-4503-1420-6. URL: <http://monaharaty.com/drupal-6.14/files/webfm/Haraty-GI2012.pdf> (cit. on p. 71).
- Harzl, Annemarie et al. (Sept. 2012). *tagstore – Formal Experiment 2011-04*. Tech. rep. Graz, Austria: Graz University of Technology. URL: <https://github.com/novoid/2011-04-tagstore-formal-experiment/> (cit. on pp. 172, 182, 183, 195).
- Hearst, Marti A. (Sept. 2000). "Next Generation Web Search: Setting Our Sites." In: *IEEE Bulletin of the Technical Committee on Data Engineering* 23.3, pp. 38–48. URL: <http://sites.computer.org/debull/A00SEP-CD.pdf> (cit. on p. 60).

Bibliography

- Hearst, Marti A. (2006). "Design Recommendations for Hierarchical Faceted Search Interfaces." In: *ACM SIGIR Workshop on Faceted Search*, pp. 1–5. URL: <http://flamenco.berkeley.edu/papers/faceted-workshop06.pdf> (cit. on p. 60).
- Hearst, Marti A. (Oct. 2009). *Search User Interfaces*. Cambridge University Press. ISBN: 0521113792 (cit. on pp. 59, 60, 66, 233).
- Hearst, Marti A. (Nov. 2011). "'Natural' Search User Interfaces." In: *Communications of the ACM* 54.11, pp. 60–67. DOI: 10.1145/2018396.2018414. URL: <http://cacm.acm.org/magazines/2011/11/138216-natural-search-user-interfaces/fulltext> (cit. on p. 66).
- Hearst, Marti A. et al. (Sept. 2002). "Finding the Flow in Web Site Search." In: *Communications of the ACM* 45.9, pp. 42–49. DOI: 10.1145/567498.567525. URL: <http://flamenco.berkeley.edu/papers/cacm02.pdf> (cit. on p. 60).
- Helic, Denis et al. (Aug. 2010). "On the Navigability of Social Tagging Systems." In: *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010*. Ed. by Ahmed K. Elmagarmid and Divyakant Agrawal. Minneapolis, Minnesota, USA: IEEE Computer Society, pp. 161–168. ISBN: 978-0-7695-4211-9. DOI: 10.1109/SocialCom.2010.31 (cit. on p. 87).
- Helic, Denis et al. (2011). "Are Tag Clouds Useful for Navigation? A Network-Theoretic Analysis." In: *International Journal of Social Computing and Cyber-Physical Systems* 1.1, pp. 33–55. DOI: 10.1504/IJSCCPS.2011.043603 (cit. on p. 87).
- Houben, Steven (Nov. 2011). "Activity Theory Applied to the Desktop Metaphor." MA thesis. Diepenbeek, Belgium: Hasselt University. URL: <http://itu.dk/people/shou/pubs/StevenHoubenMasterThesis.pdf> (cit. on p. 82).
- Hsieh, J. L. et al. (Apr. 2008). "A Web-based Tagging Tool for Organizing Personal Documents on PCs." In: *International Conference of Computer-Human Interaction 2008 (CHI2008)*. Florence, Italy. URL: <http://works.bepress.com/lucemia/18/> (cit. on p. 138).
- Hurst, Mark (2007). *Bit Literacy: Productivity in the Age of Information and E-mail Overload*. 1st. New York, USA: Good Experience. ISBN: 0979368103 (cit. on p. 72).

- Huynh, David, David R. Karger, and Dennis Quan (May 7, 2002). "Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF." In: *Proc. International Workshop on the Semantic Web (WWW 2002)*. Hawaii, USA. URL: <http://semanticweb2002.aifb.uni-karlsruhe.de/proceedings/Research/huynh.pdf> (cit. on p. 29).
- Huynh, David, David R. Karger, Dennis Quan, and Vineet Sinha (Jan. 2003). "Haystack: A Platform for Creating, Organizing and Visualizing Semistructured Information." In: *Proc. 8th International Conference on Intelligent User Interfaces (IUI '03)*. Miami, Florida, USA: ACM, pp. 323–323. ISBN: 1581135866. DOI: [10.1145/604045.604116](https://doi.org/10.1145/604045.604116) (cit. on p. 29).
- Jarrett, D. and Philips Business Systems (1982). *The Electronic Office: a Management Guide to the Office of the Future*. Gower, with Philips Business Systems. ISBN: 0566034093 (cit. on p. 64).
- Johnson, Jeff et al. (1989). "The XEROX Star: A Retrospective." In: *Computer* 22.9, pp. 11–26. URL: https://wiki.cs.umd.edu/cmsc434_f09/images/c/cb/Xerox.pdf (cit. on p. 5).
- Jones, William P. (Nov. 15, 2007). *Keeping Found Things Found: The Study and Practice of Personal Information Management*. Morgan Kaufmann. ISBN: 0123708664 (cit. on pp. 67, 68, 72, 236).
- Jones, William P. (Mar. 2012). *The Future of Personal Information Management, Part I: Our Information, Always and Forever*. Morgan & Claypool. DOI: [10.2200/S00411ED1V01Y201203ICR021](https://doi.org/10.2200/S00411ED1V01Y201203ICR021). URL: <http://www.morganclaypool.com/doi/abs/10.2200/S00411ED1V01Y201203ICR021> (cit. on pp. 67, 237).
- Jones, William P., Harry Bruce, et al. (Nov. 2009). "Providing for Paper, Place and People in Personal Projects." In: *PIM09 ASIS&T 2009 Workshop, Vancouver, BC, Canada*. Vancouver, BC, Canada: ACM. URL: <http://pimworkshop.org/2009/index.php?page=acceptedpapers> (cit. on pp. 52, 71).
- Mynatt, Elizabeth D. et al., eds. (Apr. 2010). *Planz to Put Our Digital Information in its Place*. Atlanta, Georgia, USA, pp. 2803–2812. ISBN: 978-1-60558-930-5. DOI: [10.1145/1753846.1753866](https://doi.org/10.1145/1753846.1753866) (cit. on pp. 54, 89).
- Jones, William P., Predrag Klasnja, et al. (Apr. 2008). "The Personal Project Planner: Planning to Organize Personal Information." In: *International Conference of Computer-Human Interaction 2008 (CHI2008)*. Florence, Italy. Pp. 681–684 (cit. on p. 53).

Bibliography

- Jones, William P., Ammy J. Phuwanartnurak, et al. (Apr. 2005). "Don't Take My Folders Away! Organizing Personal Information to Get Things Done." In: *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*. Portland, Oregon, USA: ACM Press, pp. 1505–1508. DOI: [dx.doi.org/10.1145/1056808.1056952](https://doi.org/10.1145/1056808.1056952) (cit. on p. 51).
- Jones, William P. and Jaime Teevan (2007). *Personal Information Management*. University of Washington Press. ISBN: 9780295987378 (cit. on pp. 9, 31, 67, 234).
- Jr., Dan R. Olsen et al., eds. (Apr. 2009). *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI2009*. ACM. Boston, MA, USA: ACM. ISBN: 978-1-60558-246-7.
- Kaptelinin, Victor (Apr. 2003). "UMEA: Translating Interaction Histories Into Project Contexts." In: *CHI*. Ed. by Gilbert Cockton and Panu Korhonen. Ft. Lauderdale, Florida, USA, pp. 353–360. ISBN: 1-58113-630-7. DOI: [10.1145/642611.642673](https://doi.org/10.1145/642611.642673) (cit. on p. 53).
- Kelly, Liadh, Daragh Byrne, and Gareth J.F. Jones (Nov. 2009). "The Role Of Places And Spaces In Lifelog Retrieval." In: *PIM09 ASIS&T 2009 Workshop, Vancouver, BC, Canada*. Vancouver, BC, Canada: ACM. URL: <http://pimworkshop.org/2009/index.php?page=acceptedpapers> (cit. on p. 44).
- Kim, Jinyoung (Oct. 2011). "Evaluating an Associative Browsing Model by Simulation." In: *The Fifth Workshop on Human-Computer Interaction and Information Retrieval (HCIR)*. Mountain View, California, USA (cit. on pp. 57, 100, 231).
- Kim, Jinyoung et al. (2010). "Building a Semantic Representation for Personal Information." In: *CIKM*. Ed. by Jimmy Huang et al. ACM, pp. 1741–1744. ISBN: 978-1-4503-0099-5. DOI: [10.1145/1871437.1871718](https://doi.org/10.1145/1871437.1871718). URL: <http://people.cs.umass.edu/~abakalov/papers/cikm10-pim.pdf> (cit. on pp. 55, 56, 58, 87, 231).
- Körner, Christian et al. (June 2010). "Of Categorizers and Describers: An Evaluation of Quantitative Measures for Tagging Motivation." In: *Proc. 21st ACM Conference on Hypertext and Hypermedia (Hypertext 2010)*. Toronto, Ontario, Canada: ACM, pp. 157–166. ISBN: 1450300413. DOI: [10.1145/1810617.1810645](https://doi.org/10.1145/1810617.1810645). URL: http://kmi.tugraz.at/staff/markus/documents/2010_HT2010_Categorizers_Describers.pdf (cit. on pp. 70, 232).

- Kreyszig, Erwin (1993). *Advanced Engineering Mathematics*. 7th. Wayne Anderson. ISBN: 0-471-59989-1 (cit. on p. 95).
- Lane, David M. et al. (2005). "Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts." In: *International Journal of Human-Computer Interaction* 18.1, pp. 133–144. URL: http://www.tandfonline.com/doi/abs/10.1207/s15327590ijhc1802_1 (cit. on pp. 70, 88).
- Lansdale, Mark W. (Mar. 1988). "The Psychology of Personal Information Management." In: *Applied Ergonomics* 19.1, pp. 55–66. ISSN: 0003-6870. DOI: 10.1016/0003-6870(88)90199-8. URL: <http://simson.net/ref/1988/Lansdale88.pdf> (cit. on pp. 13, 16, 66, 73, 81, 84, 85, 88, 93, 100).
- Leung, Andrew W. et al. (2008). "Measurement and Analysis of Large-Scale Network File System Workloads." In: *USENIX Annual Technical Conference*. Ed. by Rebecca Isaacs and Yuanyuan Zhou. Boston, MA, USA: USENIX Association, pp. 213–226. ISBN: 978-1-931971-59-1. URL: http://www.usenix.org/events/usenix08/tech/full_papers/leung/leung.pdf (cit. on pp. 35, 66, 104).
- Lewis, James R. and Jeff Sauro (2012). *Quantifying the user Experience: Practical Statistics for User Research*. Denver, Colorado, USA: Measuring Usability LLC. ISBN: 9781470025571 (cit. on p. 161).
- Malhotra, Jyoti, Prachi Sarode, and Aparna Kamble (Apr. 2012). "A Review of various techniques and approaches of Data Deduplication." In: *International Journal of Engineering Practices* 1.1. ISSN: 2277-9701 (cit. on p. 79).
- Malone, Thomas W. (Jan. 1983). "How do People Organize Their Desks?: Implications for the Design of Office Information Systems." In: *Transactions on Information Systems* 1.1, pp. 99–112. ISSN: 1046-8188. DOI: 10.1145/357423.357430 (cit. on pp. 12, 19, 51, 100, 113, 234).
- Marsden, Gary and David E. Cairns (Sept. 2003). "Improving the Usability of the Hierarchical File System." In: *Proc. Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology (SAICSIT 2003)*. Fourways, RSA: South African Institute for Computer Scientists and Information Technologists (SAICSIT), pp. 122–129. ISBN: 1581137745. URL: <http://pubs.cs.uct.ac.za/archive/00000190/01/saicsit2003-dec.pdf> (cit. on pp. 41, 42).

Bibliography

- Mathis, Lukas (Aug. 2011). *Designed for Use – Create Usable Interfaces for Applications and the Web*. Ed. by Jill Steinberg. 2nd. The Pragmatic Programmers (cit. on p. 142).
- McGrath, Joseph E. (Apr. 1995). "Methodology Matters: Doing Research in the Behavioral and Social Sciences." In: *Human-Computer Interaction: Toward the Year 2000*. Ed. by Ronald M. Baecker et al. Second. Morgan Kaufmann. Chap. 2, pp. 152–169. ISBN: 1558602461. URL: <http://www.ufpa.br/cdesouza/teaching/es/2000-mcgrath.pdf> (cit. on p. 9).
- Mohan, Prashanth, Venkateswaran S Raghuraman, and Arul Siromoney (Dec. 2006). *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 (cit. on pp. 46, 47, 96).
- Morville, Peter and Lou Rosenfeld (2006). *Information Architecture for the World Wide Web*. 3rd. O'Reilly. ISBN: 978-0-596-52734-1 (cit. on p. 68).
- Nielsen, Jakob (Feb. 1996). *The Death of File Systems*. URL: <http://www.useit.com/papers/filedeath.html> (cit. on pp. 82, 100).
- Orlowski, Andrew (Mar. 2002). *Windows on a database – sliced and diced by BeOS vets*. Interview with Dominic Giampaolo and Benoit Schillings of Be (cit. on p. 86).
- Pak, Richard, Steven Pautz, and Rebecca Iden (2007). "Information Organization and Retrieval: A Comparison of Taxonomical and Tagging Systems." In: *Cognitive Technology* 12.1, pp. 31–44. URL: <http://business.clemson.edu/Catlab/pubs/pak-pautz-iden-2007.pdf> (cit. on p. 138).
- Peres, S. Camille et al. (Sept. 2004). "Keyboard shortcut usage: The roles of social factors and computer experience." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 48. 5. SAGE Publications, pp. 803–807. URL: <http://pro.sagepub.com/content/48/5/803.short> (cit. on pp. 70, 73).
- Perugini, Saverio (2009). "Supporting Multiple Paths to Objects in Information Hierarchies: Faceted Classification, Faceted Search, and Symbolic Links." In: *Information Processing and Management* 46.1, pp. 22–43. DOI: 10.1016/j.ipm.2009.06.007 (cit. on pp. 59, 82, 100, 238).
- Peters, Isabella and Katrin Weller (Sept. 2008). "Tag Gardening for Folksonomy Enrichment and Maintenance." In: *Webology* 5.3. URL: <http://www.webology.ir/2008/v5n3/a58.html> (cit. on p. 120).

- Pirrer, Michael (2012). *Implementing a Help System for tagstore*. Tech. rep. Graz, Austria: Graz University of Technology (cit. on p. 130).
- Proceedings of the Personal Information Management Workshop at the ASIS&T 2009 (PIM2009)* (Nov. 2009). Vancouver, BC, Canada: ACM. URL: <http://pimworkshop.org/2009/>.
- Quan, Dennis et al. (Sept. 2003). "User Interfaces for Supporting Multiple Categorization." In: *Proc. 9th IFIP TC13 International Conference on Human-Computer Interaction (INTERACT '03)*. Zurich, Switzerland: IOS Press, pp. 228–235. ISBN: 1586033638. URL: <http://www.idemployee.id.tue.nl/g.w.m.rauterberg/conferences/INTERACT2003/INTERACT2003-p228.pdf> (cit. on p. 100).
- Ranganathan, Sirkali Ramamrita (1933). *Colon Classification*. Madras Library Association (cit. on p. 59).
- Reinecke, Katharina and Abraham Bernstein (June 2011). "Improving Performance, Perceived Usability, and Aesthetics with Culturally Adaptive User Interfaces." In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 18.2. DOI: 10.1145/1970378.1970382. URL: http://people.seas.harvard.edu/~reinecke/Publications_files/ReineckeBernstein_ToCHI.pdf (cit. on p. 69).
- Rekimoto, Jun (Nov. 1999). "Time-Machine Computing: A Time-centric Approach for the Information Environment. Interaction Laboratory." In: *Proc. 12th Annual ACM Symposium on User Interface Software and Technology (UIST'99)*. Asheville, North Carolina, USA: ACM. URL: <http://www.sonycs1.co.jp/person/rekimoto/papers/uist99.pdf> (cit. on p. 25).
- Rhodes, Bradley J. (May 2000). "Just-In-Time Information Retrieval." PhD thesis. Boston/MA, USA: MIT Media Lab (cit. on p. 26).
- Rhodes, Bradley J. (2003). "Using Physical Context for Just-in-Time Information Retrieval." In: *IEEE Transactions on Computers* 52, pp. 1011–1014. ISSN: 0018-9340. DOI: 10.1109/TC.2003.1223636 (cit. on p. 26).
- Rhodes, Bradley J. and P. Maes (2000). "Just-in-time information retrieval agents." In: *IBM Systems Journal* 39.3.4, pp. 685–704. ISSN: 0018-8670. DOI: 10.1147/sj.393.0685 (cit. on pp. 26, 27).
- Rhodes, Bradley J. and Thad Starner (Apr. 1996). "The Remembrance Agent: A continuously running automated information retrieval system." In: *The Proceedings of The First International Conference on The Practical Appli-*

Bibliography

- cation of Intelligent Agents and Multi Agent Technology (PAAM '96), London, UK, pp. 487–495 (cit. on p. 26).
- Rodden, Kerry and Michael Leggett (Apr. 2010). “Best of Both Worlds: Improving Gmail Labels with the Affordances of Folders.” In: *Proc. SIGCHI Conference on Human Factors in Computing Systems (CHI 2010)*. Atlanta, Georgia, USA. DOI: [10.1145/1753846.1754199](https://doi.org/10.1145/1753846.1754199). URL: <http://portal.acm.org/citation.cfm?id=1754199> (cit. on p. 86).
- Rodden, Kerry and Kenneth R. Wood (Apr. 2003). “How do people manage their digital photographs?” In: *Proceedings of the 2003 Conference on Human Factors in Computing Systems, CHI2003*. Ed. by Gilbert Cockton and Panu Korhonen. Ft. Lauderdale, Florida, USA, pp. 409–416. ISBN: 1-58113-630-7. DOI: [10.1145/642611.642682](https://doi.org/10.1145/642611.642682) (cit. on pp. 55, 82).
- Rubin, Jeffrey and Dana Chisnell (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. second. Wiley. ISBN: 978-0470-386-088 (cit. on p. 142).
- Sauermann, Leo and Dominik Heim (Oct. 2008). “Evaluating Long-Term Use of the Gnows Semantic Desktop for PIM.” In: *International Semantic Web Conference*. Ed. by Amit P. Sheth et al. Vol. 5318. Lecture Notes in Computer Science. Karlsruhe, Germany: Springer, pp. 467–482. ISBN: 978-3-540-88563-4. DOI: [10.1007/978-3-540-88564-1_30](https://doi.org/10.1007/978-3-540-88564-1_30). URL: <http://www.springerlink.com/index/C7U686MQ27751373.pdf> (cit. on pp. 40, 55, 70, 89).
- Schober, Georg (2012). *A Recommender System for Tagging Files and Folders using tagstore*. Tech. rep. Graz, Austria: Graz University of Technology (cit. on p. 115).
- Schulte, Eric and Dan Davison (June 2011). “Active Documents with Org-Mode.” In: *Computing in Science Engineering* 13.3, pp. 66–73. ISSN: 1521-9615. DOI: [10.1109/MCSE.2011.41](https://doi.org/10.1109/MCSE.2011.41) (cit. on p. 53).
- Schulte, Eric, Dan Davison, et al. (Jan. 25, 2012). “A Multi-Language Computing Environment for Literate Programming and Reproducible Research.” In: *Journal of Statistical Software* 46.3, pp. 1–24. ISSN: 1548-7660. URL: <http://www.jstatsoft.org/v46/i03> (cit. on p. 53).
- Seltzer, Margo and Nicholas Murphy (May 2009). “Hierarchical File Systems Are Dead.” In: *Proceedings of the 12th Workshop on Hot Topics in Operating Systems HOTOS09*. Monte Verita, Switzerland (cit. on pp. 47, 48, 82).

- Sheth, Amit P. et al., eds. (Oct. 2008). *The 7th International Semantic Web Conference, ISWC 2008*. Vol. 5318. Lecture Notes in Computer Science. Karlsruhe, Germany: Springer. ISBN: 978-3-540-88563-4.
- Shirky, Clay (2005). *Ontology is Overrated: Categories, Links and Tags*. URL: http://www.shirky.com/writings/ontology_overrated.html (cit. on pp. 82, 85, 100).
- Sinha, Vineet and David R. Karger (2005). "Magnet: Supporting Navigation in Semistructured Data Environments." In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD '05)*. Baltimore, Maryland: ACM Press, pp. 97–106. ISBN: 1-59593-060-4. DOI: 10.1145/1066157.1066169. URL: <http://dl.acm.org/citation.cfm?id=1066169> (cit. on pp. 29, 30).
- Smith, W.R. (Feb. 1994). "The Newton application architecture." In: *Compton Spring '94, Digest of Papers*. San Francisco, CA, USA, pp. 156–161. ISBN: 0-8186-5380-9. DOI: 10.1109/CMPCON.1994.282931 (cit. on p. 208).
- Solskinnsbakk, Geir and Jon Gulla (2010). "A Hybrid Approach to Constructing Tag Hierarchies." In: *On the Move to Meaningful Internet Systems, OTM 2010*. Ed. by Robert Meersman, Tharam Dillon, and Pilar Herrero. Vol. 6427. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 975–982. DOI: 10.1007/978-3-642-16949-6_22 (cit. on p. 87).
- Sutherland, Ivan Edward (Jan. 1963a). "Sketchpad, a Man-Machine Graphical Communication System." PhD thesis. Massachusetts, USA: Massachusetts Institute Of Technology (cit. on p. 21).
- Sutherland, Ivan Edward (Jan. 1963b). *Sketchpad, a Man-Machine Graphical Communication System*. Tech. rep. 247. based on Sutherland1963 (PhD). Massachusetts, USA: Massachusetts Institute Of Technology (cit. on p. 21).
- Teevan, Jaime et al. (2004). "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. Vienna, Austria: ACM, pp. 415–422. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985745. URL: <http://people.csail.mit.edu/teevan/work/publications/papers/chi04.pdf> (cit. on pp. 83, 237).
- Thacker, C. P. et al. (1979). *Alto: A Personal Computer*. CSL-79-11. Palo Alto, CA, USA: XEROX Palo Alto Research Center. URL: <http://www.computer->

Bibliography

- refuge.org/bitsavers/pdf/xerox/parc/techReports/CSL-79-11_Alto_A_Personal_Computer.pdf (cit. on pp. 3, 5).
- Tognazzini, Bruce (Dec. 2011). *Browse vs. Search: Which Deserves to Go?* URL: <http://www.asktog.com/columns/085BrowseVsSearch.html> (cit. on p. 70).
- Trattner, Christoph et al. (2012). "Exploring the Differences and Similarities between Hierarchical Decentralized Search and Human Navigation in Information Networks." In: *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*. iKNOW '12. Graz, Austria: ACM (cit. on p. 101).
- Tunkelang, Daniel (2009). *Faceted Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers. DOI: 10.2200/S00190ED1V01Y200904ICR005. URL: <http://www.morganclaypool.com/toc/icr/1/1> (cit. on pp. 59–61, 233).
- Voida, Stephen and Saul Greenberg (Apr. 2009). "WikiFolders: Augmenting the Display of Folders to Better Convey the Meaning of Files." In: *CHI*. Ed. by Dan R. Olsen Jr. et al. ACM. Boston, MA, USA: ACM, pp. 1679–1682. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518959 (cit. on p. 53).
- Voida, Stephen and Elizabeth D. Mynatt (Apr. 2009). "It Feels Better than Filing: Everyday Work Experiences in an Activity-Based Computing System." In: *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI2009*. Ed. by Dan R. Olsen Jr. et al. ACM. Boston, MA, USA: ACM, pp. 259–268. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518744 (cit. on p. 53).
- Voit, Karl, Keith Andrews, and Wolfgang Slany (Nov. 2009). "Why Personal Information Management (PIM) Technologies Are Not Widespread." In: *PIM09 ASIS&T 2009 Workshop, Vancouver, BC, Canada*. Vancouver, BC, Canada: ACM, pp. 60–64. URL: <http://pimworkshop.org/2009/index.php?page=acceptedpapers> (cit. on pp. 75, 118, 209).
- Voit, Karl, Keith Andrews, and Wolfgang Slany (Nov. 2011). "TagTree: Storing and Re-finding Files Using Tags." In: *Proc. 7th Conference of the Austrian Computer Society Workgroup: Human-Computer Interaction (Usab 2011)*. Vol. 7058. LNCS. Graz, Austria: Springer, pp. 471–481. ISBN: 364-225-3636. DOI: 10.1007/978-3-642-25364-5_33 (cit. on pp. 96, 107, 210).

- Voit, Karl, Keith Andrews, and Wolfgang Slany (Feb. 2012a). "Creating a Comparative Environment for PIM Evaluation." In: *PIM12 CSCW 2012 Workshop*. Seattle, WA, USA. URL: <http://pimworkshop.org/2012/papers> (cit. on pp. 76, 107, 211).
- Voit, Karl, Keith Andrews, and Wolfgang Slany (May 2012b). "Tagging Might Not Be Slower Than Filing in Folders." In: *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems, Extended Abstracts (CHI2012)*. CHI '12. Austin, Texas, USA: ACM, pp. 2063–2068. ISBN: 978-1-4503-1016-1. DOI: 10.1145/2212776.2223753. URL: <http://dl.acm.org/citation.cfm?id=2212776.2223753> (cit. on pp. 145, 211).
- Voit, Karl, Keith Andrews, Wolfgang Wintersteller, et al. (Mar. 2011). "Tag-Tree: Exploring Tag-Based Navigational Structures." In: *12th Internationalen Symposium der Informationswissenschaft (ISI)*. Ed. by Joachim Griesbaum, Thomas Mandl, and Christa Womser-Hacker. Vol. 58. Information und Wissen: global, sozial und frei? Hildesheim, Germany: Werner Hülsbusch, pp. 516–518. ISBN: 978-3-940317-91-9. URL: <http://www.vwh-verlag.de/vwh/?p=620> (cit. on pp. 107, 210).
- Weinberger, David (2007). *Everything Is Miscellaneous: The Power of the New Digital Disorder*. Times Books. ISBN: 0-8050-8043-0 (cit. on pp. 80, 82, 85, 100).
- Whittaker, Steve, Tara Matthews, et al. (May 2011). "Am I wasting my time organizing email?: a study of email refinding." In: *Proceedings of the 2011 annual conference on Human factors in computing systems*. CHI '11. Vancouver, BC, Canada: ACM, pp. 3449–3458. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979457. URL: http://people.ucsc.edu/~swhittak/papers/chi2011_refinding_email_camera_ready.pdf (cit. on pp. 40, 89).
- Whittaker, Steve and Candace Sidner (1996). "Email overload: exploring personal information management of email." In: *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*. CHI '96. Vancouver, British Columbia, Canada: ACM, pp. 276–283. ISBN: 0-89791-777-4. DOI: 10.1145/238386.238530 (cit. on pp. 19, 72, 113, 234).
- Yee, Ka-Ping et al. (Apr. 2003). "Faceted Metadata for Image Search and Browsing." In: *CHI*. Ed. by Gilbert Cockton and Panu Korhonen. Ft. Lauderdale, Florida, USA, pp. 401–408. ISBN: 1-58113-630-7. DOI: 10.

Bibliography

1145/642611.642681. URL: <http://kevinli.net/flamenco-chi03.pdf>
(cit. on p. 60).

Glossary

absolute path Absolute paths are paths that contain a system-unique [path](#) starting with the system root¹

associative navigation When a person navigates through a [strict hierarchy](#), she has to remember at which location within the hierarchy she stored an [item](#). With associative navigation, she does not need to remember parts of the storage path: she only selects (sub-)categories upon association to the item she is looking for. The more selections she defines, the more detailed the implicit query gets, that results in a finer set of results. This method reflects the dominant remembering method of our brain: associations between things. See also J. Kim et al., [2010](#) and J. Kim, [2011](#).

broken link [Symbolic links](#) or [shortcuts](#) point to an [item](#) either using an absolute or a relative path. When this path changes for the original item, symbolic links or shortcuts to this item still point to the old path. This leads to broken links.

categorizer see [describer](#)

controlled vocabulary When a person is only able to use a pre-defined set of tags to tag an [item](#), this pre-defined set is called a controlled vocabulary (cv).

By using controlled vocabularies, users are able to reduce problems with homonyms, synonyms, or singular/plural: when a person is using “university” in her cv, she is not able to tag something with “uni”, “education”, “universities” or similar by mistake.

CSV comma-separated values

CV controlled vocabulary

1. With Microsoft Windows, the system root is one of the drive letter characters. On all other operating systems, the system root is (the first) forward slash (/).

describer Körner et al., 2010 presents the concept of describer and categorizer. These two behavioral patterns reflect the two extreme sides of a spectrum. On the one side, describing behavior is when people are typically using many tags per item, using homonyms and synonyms as additional tags, and using words from the actual content as tags as well. On the other side, categorizing behavior typically stands for people who are using less tags per item, using tags to put the item in a higher context, and therefore some tags are not even part of the item content at all.

In this thesis, I embrace this concept of user behavior and re-uses this concept slightly different: instead of *user* who has a certain tendency towards describing or categorizing behavior, there are *tags* that reflect a certain tendency for being a descriptive tag or for being a categorizing tag. In order to do research with this concept, test users are being confronted with two distinct lines for tagging: in one tag line users are asked to put in descriptive tags and in the other tag line users are asked to find tags that categorize the item.

desktop metaphor Alan Kay first introduced the desktop metaphor while working at Xerox [Palo Alto Research Center \(PARC\)](#) in 1970. Before that, computers could only be used by special trained persons. The common interface was text-based where the user had to remember computer commands. With the development of the [Graphical User Interface \(GUI\)](#), even less trained persons were able to interact with a computer. In order to help those untrained persons even more, the desktop metaphor introduces concepts of already known real world elements to the virtual world of the computer. Display elements such as a desktop surface (the screen surface), files (paper files), folders (paper file folders), drawers (remote network computer folders), and printer icons helped users to transfer interaction knowledge from the real world to the virtual world (see [Figure 1.2](#), page 5). Therefore, even novice computer users could guess that dropping a file onto a folder leads to a file movement to this folder.

desktop search In the last decade, numerous software product solutions were developed to crawl and index local data pools. Desktop search engines provide a [search](#) interface in order to find local information.

DFA deterministic finite automaton

directory In early computer file systems, files and sub-directories were

combined within directories forming a hierarchically-structured file system. With the development of the desktop metaphor, **GUI** systems started to use the name folders for directories to emphasize the metaphor of a physical paper folder. **GUI** operating systems of the 1970s and Apple's operating systems were using the term folder. Microsoft used the term directory and changed to the term folder with the introduction of Windows 95.

Therefore, directories are a concept of the file system and folders are a concept of the user interface.

ext2fs The ext2 or second extended filesystem is a **file system** for the Linux kernel.²

faceted navigation see **faceted search**

faceted search In Hearst, 2009, p.188f and Tunkelang, 2009, the authors describe a technique named "Faceted Navigation" or "Faceted Search". This technique has properties from **search** as well as from **navigation** and can be seen as an in-between method.

Values like "nine", "ten", "black", and "brown" are associated to distinct categories (facets) like "shoe size" and "shoe color". The user interface allows selection of values for each facet like "shoe size: nine" and "color: black". The result set is filtered accordingly. For details, see Section 2.2.20 on page 59.

FAQ Frequently Asked Questions

file A file is the smallest (user-relevant) information-containing entity within a file system. End user applications store their data in files that have special file formats. The concept of files is an invention of the 1950s.

file extension A file extension reflects the internal file format. By convention, the last part of a file name consists of a dot followed by a (usually) three letter abbreviation.

For example the file Letter to John.pdf (with its file extension .pdf) is usually a file in **portable document format (PDF)** file format defined by the company Adobe. Modern operating systems use file extensions to determine which user application to use for opening a certain file.

file system Current operating systems³ use traditional file systems like

2. <https://en.wikipedia.org/wiki/Ext2fs> – retrieved on 2012-08-15.

3. Microsoft Windows 7, Mac OS X 10.7, Linux v3.4

Glossary

New Technology File System (NTFS), Hierarchical File System Plus (HFS), (V)FAT, or ext2fs to store and manage system and user files. Common to all of those file systems is an inode management structure using a strict hierarchy. This thesis proposes a different kind of management principle for system files and user files. In this document, the term file system is used in the traditional way. See also W. P. Jones and Teevan, 2007, p. 137f.

filer Inspired by the famous Malone, 1983 paper, many studies like Whittaker and Sidner, 1996 or Alvarado et al., 2003 adopted the terms “filer” and “piler” to classify user behavior. People who show filer behavior tend to organize things neatly in well-developed categories, have less clutter in their working space, and few emails in their inbox. Pilers tend to stack documents and things into piles. Each pile may have a special meaning to the user, like important things, urgent things, things that might be important.

folder A computer folder is a virtual container of one or more files and one or more sub-folders in a GUI. A folder may reflect the content of a directory of the file system. Virtual folders reflect the results of search queries or other dynamically generated sets of files and folders.

folder hierarchy File systems are designed to hold a strict hierarchy of directories containing other directories and/or files (see Figure 3.1, page 78). Folder hierarchies were developed in the 1960s (see Freeman and Gelernter, 1996).

GPL GNU General Public License

Graphical User Interface In contrast to text-based computer interfaces, GUI-based computer interfaces allow graphical elements such as images to interact with a user (see Figure 1.2, page Figure 5). Display elements can be manipulated directly, typically by using a mouse, a computer pointing device. In recent years, touch screen systems which allow screen element manipulation with one or more fingertips (multi-touch) have become popular.

GTD Getting Things Done

GUI Graphical User Interface

hard link Most file systems implement hard links in a way that they are alternative representations of an item. Unlike symbolic links or shortcuts which are links to absolute or relative paths (item names), hard

links represent an alternative item name to the identical item.

There can not be a broken hard link: when one item name gets re-named, all other hard links referring to the same file stay untouched and working. Only when the last hard link that links to an item gets deleted, the item gets deleted.

HDD Hard Disk Drive

HFS Hierarchical File System Plus

inode The smallest file- or directory-related information entity managed by a file system. Usually, a file is stored in a set of linear linked inodes. A link or a directory consists of a single inode.

The size of an inode is defined by the so called chunk size of the file system which is set-up while being instantiated. Typically, an inode has 4.096 Bytes. A file system is only able to store files in multiples of inode size quantities. This means that a file which stores 42 Bytes uses up 4.096 Bytes on the hard disk. The difference of 4.052 Bytes is part of the “internal fragmentation”. In extreme cases this could lead to a full partition with stored data of a fraction of the partition capacity. Most file systems have a hard limit on the number of inodes.⁴

item In this thesis, the term “item” refers to either a [file](#) or a [folder](#).

KISS keep it short and simple

link In this thesis the term link refers to either a [symbolic link](#), a [hard link](#), or a [shortcut](#). In context of the tagstore software, a link refers to either a symbolic link (Linux, Mac os x) or a shortcut (Microsoft Windows).

metadata Data describing the content or the structure of other data is called metadata. Baeza-Yates and Ribeiro-Neto (2011, p. 205) define metadata as “information on the organization of the data, the various data domains, and the relationship between them. In short, metadata is ‘data about the data’.”

MVC Model-View-Controller

navigation A person who inspects contents of (sub-)folders in a software interface that visualizes (only parts of) the file system hierarchy is using the navigation method. Re-finding information using navigation

⁴. There are some exceptions to this fact. For example in [Extended File System \(xfs\)](#), inodes are assigned dynamically and therefore can hold much more entities.

Glossary

relates to recognition. In contrast to [search](#), no queries have to be entered explicitly.

An exhaustive collection on definitions related to the terms “navigation” or “browsing”⁵ can be found in **Rice2001** Baeza-Yates and Ribeiro-Neto, [2011](#), p. 24, and W. P. Jones, [2007](#), p. 93ff. This thesis does not limit user behavior to orienteering, random browsing, directed, undirected, semidirected, or undirected browsing only. The subject of this work is not the reasoning about the motivation of a person who uses tools for navigation. Although these more elaborate definitions might get important for interpreting the results.

NTFS New Technology File System

PARC Palo Alto Research Center

partition Physical hard disks are divided into smaller logical entities which are called partitions.

In Microsoft operating systems, partitions are addressed using drive letters like C:. In all other major operating systems, drives are mapped to directories. These mapping directories are called “mount points”. Historically the division of a hard drive into partitions was necessary because early operating systems could not address the large memory of a single drive in one logical entity. Modern operating systems are able to address even very large hard drives⁶ in one single partition.

path A path to an item consists of a set of [directories](#) or [folders](#) and an optional [file](#). The different components of a path are separated by system-dependent path separation characters. Microsoft Windows uses backward slashes (\) and all other operating systems use forward slashes (/) as path separator characters.

PDF portable document format

piler see [filer](#)

PIM Personal Information Management

Quantified Self The Quantified Self movement was proclaimed by two online magazine editors in 2007. People practicing the Quantified Self use modern technology to collect and analyze all kinds of data which relates to them personally. It can be used to track physical shape,

⁵. The two terms are being used as synonyms throughout most papers.

⁶. Currently, the largest hard disks available are 4 Terabytes.

health issues, or generate personal behavior statistics just for the fun of doing it.

QUIS Questionnaire for User Interaction Satisfaction

RDF Resource Description Framework

relative path When an item is referred to by a relative path, the source of the path resolution is the current working [directory](#). For example, the relative path `./afile.txt` refers to the item `afile.txt` in the current directory⁷.

The relative path `.././subdir1/anotherfile.txt` points to the item `anotherfile.txt` which resides in the directory `subdir1` which itself resides two directories⁸ in the direction of the system root.

RSS Rich Site Summary

search In a search, the user formulates a search query for the search interface. The systems provides a result set and, probably, methods to filter the result set. Users scan through the result set, re-formulate a different search query, find the information they were seeking for, or start a navigation process within the preposed results. The term teleporting is sometimes used in the same manner as search (see Alvarado et al., 2003; Chau, Myers, and Faulring, 2008b) and sometimes it is differentiated (see Teevan et al., 2004, W. P. Jones, 2012, p. 18).

This thesis concentrates solely on navigation and not on search. I think that search and navigation are both methods that every user should be able to use on all kinds of information.

semantic web The semantic web denotes a highly [metadata](#)-enriched version of the Internet where semantic data technology is used to link and query information.

shortcut Microsoft Windows uses special files, which are called shortcuts, instead of [links](#). Those shortcut files are files with the hidden⁹ [file extension](#) `.lnk`. The file content contains an [absolute path](#) to the target item.

Unlike [symbolic links](#) and [hard links](#) which are part of the file system

⁷. The dot stands for the current working directory.

⁸. Two dots stand for the directory which contains the current directory.

⁹. The file extension `.lnk` is hidden independently of the setting “Hide extensions for known file types”.

Glossary

layer, shortcuts are an extension of Windows Explorer. Any software product that has to interpret shortcut files, has to implement support for them.

Due to the fact that shortcut files have to be interpreted by user-space software to be resolved, using many shortcut files may arise performance issues.

SIS Stuff I've Seen

social tagging Systems, where different persons see other persons' tags associated to information and are able to add tags by themselves are social tagging systems. Through the power of the crowd, more prominent tags evolve while infrequent or fuzzy tags allow for finding information even with vaguer search queries. Due to the nature of the method proposed, this thesis does not cover social tagging aspects.

SSD Solid State Drive

strict hierarchy A strict hierarchy is an acyclically directed graph. It is a tree without cyclic relations¹⁰, where one node is contained only in one (father-) node. Nodes can have zero to many sub-nodes.

SVN Subversion

symbolic link An [inode](#) that refers to an inode of an [item](#) is called a symbolic link (in short: symlink). With symbolic links, a user can create an alias to a file or folder that is probably located in a different folder. Microsoft Windows uses [NTFS](#) as its file system. Although [NTFS](#) provides technical representations of links, Windows does not offer any usable user-space tools to maintain links. Instead, Windows uses so-called [shortcut](#) files. All other operating systems use file systems that implement symbolic links. See also Perugini (2009, pp. 33ff).

tag cloud To visualize a set of tags and the "weight" of certain tags related to a criteria (such as the number of appearances), tag clouds are used by many web services. To emphasize tags according to the weight function, larger character sizes or color are usually used.

Many web services use tag clouds to give a quick overview of a set of tags related to a topic or a user. Due to the fact that clicking on one tag of a tag cloud simply results in a view of all items tagged with this one tag in most implementations, tag clouds can not be seen as a navigational method.

¹⁰. With exceptions of links.

In the tagstore dialog, a tag cloud is used to visualize tags proposed by the [tag recommender system](#) as shown in Figure 5.2 (page 112).

tag completion An important usability feature of any tagging dialog is tag completion. When the user starts typing the first letter, a drop-down overlay shows all tags starting with this letter. The more letters the user types, the more specific the list gets. At any time, the user can select a complete tag from the list. This has several advantages. On the one hand, users quickly see all tags that start with the sequence of characters already typed in. On the other hand, many tags do not have to be typed completely.

In the tagstore dialog screenshot of Figure 5.2 (page 112), tag completion of the tags “history” or “hifi” is shown.

tag recommender system Many tagging systems provide a tag recommendation system (or short: recommender) in order to give an educated guess, what (already known or unknown) tags the user might want to use for the current item.

In the tagstore dialog, tag recommendations are visualized using a [tag cloud](#) as shown in Figure 5.2 (page 112).

tagging The process of adding [metadata](#) (tags) like cues, keywords, or even combinations of words to items is called tagging. The set of an allowed length of characters per tag differs from implementation to implementation: some products allow only single words, while others allow arbitrary sequences of characters as tags.

TP test person

URL Uniform Resource Locator

(V)FAT The [file system](#) used in Microsoft Windows versions prior to xp. It is still used for most memory sticks, external hard disks, and so forth.

XFS Extended File System

Index

- Alan Kay, 3
- Alto, *see* Xerox Alto
- Android, 122
- Aristotle, 3
- association, 12
- associative navigation, *see* navigation, associative
- Attribute Browser, 41
- book, 3
- bookmarks, 17, 24, 32, 49, 67
- browsing, *see* navigation, 233
- CAD, 21
- categories, 16
- categorizer, 70, 229
- classification
 - automatic, 100
 - deferring, 100, 113
 - multiple, 13, 82, 86, 94, 100
 - overlap, 19, 32
 - problems, 14
 - wrong, 37
- Clay Shirky, 85
- cloud services, 4, 67
- cognitive model, 79
- collaborative, 23, 24, 68, 70
- Colon Classification, 59
- computer mouse, 22
- controlled vocabulary, 71, 105, 116, 229
- copy and paste, 22
- costs, 79, 84
- cross-tool behavior, 19
- David Gelernter, 23
- David Weinberger, 80, 85
- deduplication, 79
- dependent variable, 9
- describer, 70, 229
- desktop metaphor, 3, 15, 16, 77–82, 230
- desktop search, 230
- directory, 3, 23, 230
- Douglas Engelbart, 22
- Emacs, 26, 67, 208
- email, 17, 32, 33, 37, 39, 45, 49, 53, 56, 67, 68, 72
- Eric Freeman, 23
- faceted navigation, 59, 87, 231
- faceted search, 49, 56, 59, 231
- Feldspar, 49
- file system, 3, 45, 47, 64, 77, 86, 208, 231
 - semantic, 45

Index

- filename, 38
- filers, 12, 19, 189, 231
- files, 231
 - archived, 19, 64, 81
 - ephemeral, 19
 - extension, 231
 - growing numbers, 64–66, 69
 - lost, 66, 104
 - ownership, 19
 - personal relation, 18
 - usefulness, 19
 - working, 19
- Flamenco, 60
- folder, 232
 - feeling of control, 53
 - hierarchy, 20, 42, 101, 232
- GTD, 72
- GUI, 4, 69, 110, 232
- Haystack, 29
- hFAD, 47
- hierarchy, 43, 47, 51, 66, 76
 - of categories, 3
 - of folders, 232
 - problems, 77, 84
 - re-using, 32, 51, 87
 - strict, 235
- hypertext links, 22
- independent variable, 9
- information retrieval, *see* retrieval
- inode, 232
- item, 233
- Ivan Sutherland, 21
- keyboard shortcuts, 70, 88
- library systems, 16
- Lifestreams, 23
- LiFiDeA, 55
- links, 11, 43, 56, 77, 78, 86, 233
 - broken, 79, 229
 - hard, 232
 - symbolic, 112, 236
 - usage of, 79
- Marti Hearst, 59
- Memacs, 67
- memex, 11, 67
- Menlo Park, 22
- metadata, 46, 47, 55, 109, 115, 233
 - implicit, 44, 100
- movable types, 3
- Multics, 3
- My Tags, 116, 126
- MyLifeBits, 42, 67
- navigation, 6, 23, 43, 51, 82–84, 94, 233
 - as implicit query, 96, 102
 - associative, 45, 57, 58, 94, 97, 100, 132, 229
 - by facets, *see* faceted navigation, 231
 - changes in, 66, 69
 - preferred over search, 19, 82–84
 - using semantics, 46
- NLS, 22
- open source, 75, 109
- Org-mode, 67, 208
- partition, 234
- path, 234
- Personal Project Planner, *see* Planz

- Phlat, 37
- pillers, 12, 19, 189, 231
- piles, 14
- PIM
 - devices, 67
 - planning, 20, 71
 - strategies, 20, 52, 61, 66, 68, 79
- Placeless Documents, 28
- Planz, 51
- Presto, 28
- printing, 3
- priorities
 - changing, 13
- Quantified Self, 45, 73, 234
- Querium, 61
- QUIS, 25
- real world, 15, 16, 74
- Remembrance Agent, 26
- reminding, 12, 23, 25, 54, 74
- retrieval, 16, 55, 82–84
 - problems, 17, 79, 101
- search, 23, 26, 33, 40, 43, 48, 49, 58, 68, 76, 82, 85, 234
 - by facets, *see* faceted search
- semantics, 30, 45, 47, 55, 87, 235
- SemFS, 46
- SenseCam, 44
- shortcuts, 42, 112, 235
- SIS, *see* Stuff I've Seen
- Sketchpad, 21
- Smart Folders, 28
- social, 52, 73
 - pressure, 20
 - tagging, *see* tagging, social
- Star, *see* Xerox Star
- Stuff I've Seen, 33
- summarizing, 23
- tag
 - cloud, 236
 - completion, 76, 104, 166, 236
 - recommendation, 76
 - recommendations, 104, 114–115, 166, 236
- TagFS, 47
- tagging, 39, 41, 46–48, 56, 70, 76, 94, 96, 99, 104, 237
 - benefits, 104
 - optimizing, 104
 - seen as a burden, 104
 - social, 94, 235
- tagstore, 107–140
- TagTrees, 94–105, 132
 - example, 98
 - number of links, 95
 - supporting serendipity, 104
 - usage for file management, 96–99
- teleporting, 235
- Time Machine, 25
- time-oriented, 25, 86, 116
- TimeScape, 25
- TmSamba, 25
- user, 69
 - acceptance, 25
 - behavior, 57, 88, 89
- users, 105
 - learning, 70–73, 77, 105
 - motivation, 18
 - personal relation to files, 18
- Vannevar Bush, 10

Index

visio-spacial methods, [16](#)

Vista, [29](#)

Vocabulary Problem, [84](#)

WM, *see* WorkspaceMirror

WorkspaceMirror, [18](#), [32](#)

Xerox

Alto, [3](#)

Star, [5](#)

List of Figures

1.1	Xerox Alto	4
1.2	Screenshot: Xerox Star 8010 graphical user interface	5
2.1	Influence factors and feedback methods	10
2.2	TX-2 Operating Area – Sketchpad in use	21
2.3	Douglas Engelbart during his demonstration of NLS	22
2.4	Screenshot: Lifestreams interface, UNIX Viewport	24
2.5	Screenshot: TimeScape desktop	26
2.6	Screenshot: Emacs showing Remembrance Agent	27
2.7	Screenshot: Vista, a Presto interface	29
2.8	Screenshot: Haystack, Paper writing workspace	31
2.9	Screenshot: Stuff I’ve Seen (sis) interface, with the Top View.	34
2.10	Screenshot: sis interface, with the Side View.	36
2.11	Screenshot: Phlat	38
2.12	Screenshot: Feldspar	50
2.13	Screenshot: Planz	54
2.14	Screenshot: LiFiDeA	56
2.15	Screenshot: Faceted search on eBay.com	59
2.16	Screenshot: Flamenco faceted search interface	60
3.1	A dilemma with strict folder hierarchies	78
3.2	A humorous summary of not being able to organize	80
3.3	Tool optimization became stuck	91
4.1	The TagTree structure of an example file	97
4.2	Three different examples on how to navigate to an item	98
4.3	Adding to folder hierarchies compared to TagTrees	103
5.1	The tagstore implementation layers	111
5.2	tagstore dialog screenshot	112

List of Figures

5.3	tagstore Manager – My Tags	116
5.4	tagstore Manager – Datestamps	117
5.5	tagstore Manager – Expiry Date	118
5.6	tagstore Manager – Re-Tagging	119
5.7	tagstore Manager – Rename Tags	121
5.8	tagstore Manager – Store Management	122
5.9	tagstore Manager – Sync Settings	123
6.1	FE1: Example background questionnaire from TP05, page one	148
6.2	FE1: Example background questionnaire from TP05, page two	149
6.3	FE1: Example feedback form from TP05	150
6.4	FE1: Photograph of the test environment	151
6.5	FE1: Screenshot of tagstore rev. 226	153
6.6	FE1: Windows Explorer showing files and mapped drives . .	155
6.7	Data processing from video to csv result files	156
6.8	FE1: Deterministic finite automaton (DFA): filing in folders . .	157
6.9	FE1: DFA: tagging with tagstore	158
6.10	Summary of the distraction states	160
6.11	DFA for refinding	160
6.12	FE1: Filing performance for both conditions	163
6.13	FE1: Boxplot of the filing tasks	164
6.14	FE1: Re-finding performance for both conditions for file 2–6 .	165
6.15	FE1: Boxplot of the re-finding tasks	166
6.16	FE1: Filing performance within tagstore, sorted	167
6.17	FE1: Filing and re-finding performance of the fast performers	167
6.18	FE1: Boxplot: unique occurrences of folders and tags	168
6.19	FE1: Comparison of the folder and tag lengths	168
6.20	FE1: Comparison of associated number of folders and tags . .	169
6.21	FE2: Example background questionnaire from TP18, page one	174
6.22	FE2: Example background questionnaire from TP18, page two	175
6.23	FE2: Example feedback form from TP18, filing, page one . . .	177
6.24	FE2: Example feedback form from TP18, filing, page two . . .	178
6.25	FE2: Example feedback form from TP18, re-finding, page one	179
6.26	FE2: Example feedback form from TP18, re-finding, page two	180
6.27	FE2: Example feedback form from TP18, re-finding, page three	181
6.28	FE2: Screenshot: TP filing in tagstore	184
6.29	FE2: The DFA for the filing task	185

List of Figures

6.30	FE2: The DFA for the tagging task	185
6.31	FE2: Bar Chart for filing in both conditions	187
6.32	FE2: Bar Chart for re-finding in both conditions	187
6.33	FE2: Boxplot of the filing tasks	188
6.34	FE2: Boxplot of the re-finding tasks	189
6.35	FE2: Comparison of the folder and tag lengths	192
6.36	FE2: Comparison of associated number of folders and tags	193
6.37	FE2: Boxplot: unique occurrences of folders and tags	194
6.38	FE2: User feedback for filing in both conditions	195
6.39	FE2: Gender differences when filing	196
6.40	FE2: Platform differences when filing	196
6.41	FE2: Differences related to study background when filing	197
6.42	FE2: Preferred condition for filing after filing	197
6.43	FE2: Affirmation to use tagstore on computer after filing	198
6.44	FE2: User feedback for re-finding in both conditions	199
6.45	FE2: Gender differences when re-finding	199
6.46	FE2: Differences related to tagging experience when re-finding	200
6.47	FE2: Non-Windows users rating re-finding	200
6.48	FE2: Preferred condition for re-finding after re-finding	201
6.49	FE2: Affirmation to use tagstore on computer after re-finding	201

List of Tables

2.1	Cross-tool profiles in PIM	20
3.1	Statistics of my personal file system	65
4.1	Growth of the number of paths in TagTrees	96
5.1	Performance measurements TagTrees creation	136
6.1	FE1: Task performance	162
6.2	FE1: Feature usage	169
6.3	FE1: Results of the feedback questionnaire	170
6.4	FE2: User groups and test users	183
6.5	FE2: Total task times for all TPs	186
6.6	FE2: Statistical significance for filing in folders	190
6.7	FE2: Statistical significance for filing in tagstore	190
6.8	FE2: Statistical significance for re-finding in folders	191
6.9	FE2: Statistical significance for re-finding in tagstore	192