

University of Southampton Research Repository ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

UNIVERSITY OF SOUTHAMPTON

Enhancing Retrieval and Discovery of Desktop Documents

by

Gontlafetse Mosweunyane

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

December 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Gontlafetse Mosweunyane

Personal computers provide users with abilities to create, organize, store and access large amounts of information. Most of this information is in the form of documents in files organized in the hierarchical folder structures provided by the operating system. Operating system-provided access to these data is mainly through structure-guided navigation, and more recently through keyword search.

This thesis describes the author's research into the accessibility and utilization of personal documents stored and organized using the hierarchical file system provided by common operating systems. An investigation was carried out on how users currently store and access their documents in these structures. Access and utility problems triggered a need to reconsider the navigation methods currently provided. Further investigation into navigation of personal document hierarchies using semantic metadata derived from the documents was carried out. A more intuitive exploratory interface that exposes the metadata for browsing-style navigation was implemented. The underlying organization is based on a model for navigation whereby documents are represented using index terms and associations between them exposed to create a linked, similarity-based navigation structure. Exposure of metadata-derived index terms in an interface was hypothesized to reduce the user's cognitive load and enable efficient and effective retrieval while also providing cues for discovery and recognition of associations between documents.

Evaluation results of the implementation supports this hypothesis for retrieval of deeply located documents, as well as better overall effectiveness in associating and discovery of documents. The importance of semantic document metadata is also highlighted in demonstrations involving transfer of documents from the desktop to other organized document stores such as a repository.

Contents

Acknowledgements	ix
1 Introduction	1
1.1 Motivation	1
1.2 Research Context and Overview	2
1.3 Research Objectives	6
1.4 Contributions	6
1.5 Thesis Outline	7
2 Background	9
2.1 Introduction	9
2.2 Operating System-supported Organization and Access on the Desktop . .	9
2.2.1 The Desktop Metaphor and Hierarchical File Systems	9
2.2.2 Document Access on the Desktop	11
2.2.2.1 Navigating the File Hierarchy	11
2.2.2.2 Operating System-provided Search	11
2.2.2.3 Dynamic Organization and Browsing	12
2.3 Access Through File System Hierarchy Interfaces	13
2.3.1 Space Filling Techniques	13
2.3.2 The Indented List	16
2.3.3 Focus + Context Techniques	16
2.3.4 Node-Link Trees	17
2.3.5 Zoomable User Interfaces	18
2.3.6 Combination Approaches	19
2.4 Moving Away from Organization in File Hierarchies	20
2.4.1 Attribute-based Systems - Placeless Documents	20
2.4.2 Time-based Retrieval	21
2.4.2.1 Lifestreams	21
2.4.2.2 Spatial Arrangement and Time Ordering - Timescape . .	22
2.5 Connecting Information on the Desktop	23
2.5.1 Semantic Personal Information Management tools	24
2.5.1.1 SEMEX	24
2.5.1.2 Haystack	25
2.5.1.3 Gnowsis	26
2.5.1.4 IRIS	28
2.5.2 Database-based Methods	29
2.5.2.1 Iolite	29

2.5.2.2	MyLifeBits	29
2.6	Discussion	30
2.6.1	Problems with Organization and Access on the Desktop	30
2.6.2	Current Solutions and Influence on Proposed Solution	31
2.6.2.1	Hierarchy Visualizations	31
2.6.2.2	Attribute-, Time-based and Spatial Systems	32
2.6.2.3	Personal Information Integration Tools	32
2.7	Summary	32
3	A Study of Organization and Access in Personal File Hierarchies	35
3.1	Introduction	35
3.2	User Group	36
3.3	User Hierarchy Enumeration Study	36
3.4	Qualitative Study on Working with Documents on the Desktop	38
3.5	Results and Discussion	38
3.5.1	User Hierarchy Enumeration Results	38
3.5.2	Questionnaire Results	40
3.6	Summary	43
4	Problem Definition and Solution Design Issues	45
4.1	Introduction	45
4.2	Shortfalls in Aiding Retrieval and Association of Desktop Documents	46
4.2.1	Finding and Reminding on the Desktop	46
4.2.2	Relating Documents	46
4.2.3	Proposed Solutions: Browsing and Exploration Interfaces	47
4.3	Design Lessons for Information Seeking Interfaces	48
4.3.1	Information Seeking and Information Retrieval	48
4.3.2	Information-seeking Strategies	49
4.3.3	User Actions in Information Seeking	50
4.3.4	Browsing and Search on the Desktop	52
4.3.5	Facets For Organization	52
4.3.6	Document Clustering	53
4.3.7	Recommendations for Design of Personal Information Systems	54
4.3.8	Discussion	55
4.4	Document Attributes as a Basis for the Solution	56
4.4.1	Metadata	56
4.4.2	Tagging	57
4.5	Summary	59
5	Metadata Specification and Experiments	61
5.1	Introduction	61
5.2	Metadata From Desktop Documents	61
5.2.1	Microsoft Systems' File Properties	61
5.2.2	Extracting File Properties in Windows	62
5.3	Encoding and Storing the Metadata: RDF and the Semantic Web	64
5.3.1	The Semantic Web	64
5.3.2	The Resource Description Framework	64

5.3.3	Defining an Ontology	65
5.3.4	Defining URIs for the Files	66
5.3.5	The Semantic Desktop File Metadata	66
5.4	Experimenting with Specification and Filtering Using Attributes	68
5.4.1	Specification of File Metadata in Windows	68
5.4.2	Filtering Documents Using Metadata	69
5.4.3	Browsing Semantic Metadata	71
5.4.4	Adopting and Reengineering Tabulator Modules	72
5.4.5	Browsing, Query-by-example (QBE) and File-Open Interface	75
5.4.6	Adding and Editing Attributes in the Interface	77
5.4.7	Connecting to Other Semantic Information spaces	78
5.4.8	Discussion and Challenges	81
5.4.8.1	The Document Browser Architecture	81
5.4.8.2	Determining Similarity	82
5.4.8.3	Working with RDF Data	84
5.4.8.4	Querying During Browsing	85
5.4.8.5	Moving Away from Tree-like Navigation	85
5.5	Summary	85
6	The Desktop File Browser (DeFiBro)	87
6.1	Introduction	87
6.2	A Model for Desktop Document Retrieval and Association	88
6.2.1	Metadata Processing	88
6.2.1.1	Building the Document Metadata index	88
6.2.1.2	Building a Unified Metadata Index	89
6.2.2	Term-based similarity measurement	89
6.3	Metadata-Based Exploration through an Interface	93
6.4	Improving File Browsing on the Desktop	95
6.4.1	Processing the Semantic Metadata for Presentation on Interface	95
6.4.1.1	Parsing RDF Data	95
6.4.1.2	Facets for Browsing	96
6.4.1.3	Terms and Filename Indexes	96
6.4.1.4	Implementing the Conceptual Model for Browsing Documents	96
6.4.1.5	Preview Generation	97
6.4.1.6	Challenges in Building the Model	99
6.4.2	DeFiBro Architecture	99
6.4.3	The Browsing Interface	100
6.5	Integrating the Desktop with Repositories	103
6.5.1	Introduction	103
6.5.2	SWORD Deposits from the Desktop to EPrints	105
6.5.3	Comparison to Other Projects	110
6.6	Summary	110
7	Evaluation - Hierarchy Navigation vs Metadata-based Navigation	112
7.1	Introduction	112
7.2	Method of Evaluation	113

7.3	User Tasks - Context and Setup	114
7.3.1	Testing with a Test Hierarchy versus a User's Personal Hierarchy	114
7.3.2	Evaluation Tasks and Research Objectives	115
7.3.3	Test User Group	117
7.3.4	Tasks Details	117
7.4	Task Results and Analysis	119
7.4.1	Retrieval Times per Task	119
7.4.2	Hypothesis Testing by Statistical Significance Analysis of the Results	122
7.4.3	Precision and Recall for Multi-Document Task Results	125
7.4.4	Users' Approach to the Tasks and Feedback	126
7.4.4.1	Browsing Behaviour in Windows Visualization	126
7.4.4.2	Browsing Behaviour in DeFiBro	128
7.4.4.3	Questionnaire Results	130
7.5	Discussion	131
7.5.1	Improving Efficiency of Browsing and Locating Documents	131
7.5.2	Improving Effectiveness of Browsing	132
7.5.3	Other Issues	132
7.6	Summary	134
8	Concluding Remarks and Further Work	135
8.1	Summary	135
8.2	Further Work	137
8.2.1	Metadata Extraction and Presentation	137
8.2.2	Use of Extended Attributes	138
8.2.3	Browsing Google Results	138
8.2.4	Evaluation Through Undirected Browsing and Extended Use	139
8.2.5	Augmentation of Hierarchies with Context, Detail and Relations	140
8.2.6	Scalability and Seamless Integration with Hierarchy	140
8.2.7	Improving User Control and Reasoning	141
8.3	Final Remarks	141
A	Preliminary Study Information and Questionnaire	142
B	The File Ontology	146
C	User Evaluation Tasks and Questionnaire	150

List of Figures

1.1	Data integration approaches feed off data from the desktop including documents	5
2.1	The legend window, control panel and file structure viewing window of the Treemap browser (Stasko et al. (2000))	14
2.2	The Sunburst visualization of a file hierarchy (Stasko et al. (2000))	15
2.3	Browsing a WWW hierarchy using the hyperbolic browser (Lamping et al. (1995))	17
2.4	Presto system core architecture (Dourish et al. (1999a))	20
2.5	The Lifestreams interface (Freeman and Gelernter (1996))	22
2.6	Typical screenshot of Timescape (Rekimoto (1999))	23
2.7	Sample screenshot of the SEMEX interface (Cai et al. (2005))	24
2.8	SEMEX system architecture	25
2.9	Haystack showing a view of a user's inbox (Karger and Jones (2006)) . . .	27
2.10	PIMO ontology components (Sauermann et al. (2006))	27
2.11	IRIS framework showing the three-layer architecture (Cheyer et al. (2005))	29
2.12	The MyLifeBits platform store and the suite of capture/display tools (Gemmell et al. (2006))	30
3.1	Relationship between numbers of documents and files with folders they are stored in	39
3.2	Average documents stored in the number of levels	40
3.3	Number of levels used by each user	41
3.4	Number of levels used by each user having the given number of documents	41
5.1	Generic built-in file properties for a Microsoft Word file in Windows XP .	62
5.2	WMI architecture and its interaction with other components	63
5.3	Generic RDF description	65
5.4	Data model of example statement	65
5.5	Overview of the file ontology	67
5.6	An example of a RDF description for a file	68
5.7	Adding value for attribute while saving a Microsoft Word document . . .	70
5.8	Selecting Microsoft Word documents with specified metadata	71
5.9	Browsing the metadata using the Tabulator Extension 0.8.7	73
5.10	Browsing properties and values	73
5.11	Dependency of modules in Tabulator and the AJAR library (Decentralized Information Group, MIT (2006))	74
5.12	Documents metadata as resources	75
5.13	Query-by-example embedded in interface in the re-engineered browser . .	76

5.14	Selecting a date filters document list to include only relevant results	77
5.15	Browsing property values which are URIs gives more information	78
5.16	Browsing with SEMEX	79
5.17	Editing the “subject” attribute	79
5.18	Filtering documents by ECS affiliates	81
5.19	The overall system architecture	82
5.20	Wrong classification as a result of using individual words	83
5.21	The structure of a list of all degrees in ECS, visualised using Tabulator .	84
6.1	The index derivation process	90
6.2	Browsing by keywords extracted from document set	98
6.3	Selecting a document shows its keywords, preview and related documents	98
6.4	Overall Architecture of the System	100
6.5	Browsing Interface Architecture	101
6.6	Selecting the facet ‘Author’ shows the values of the facet	101
6.7	Selecting the facet value shows the documents matching the criteria . . .	102
6.8	Selecting the documents based on first letter of filename (filename index)	102
6.9	Pointing to a document name/icon reveals its preview in the preview pane	103
6.10	Selecting a document reveals the associated terms (keywords) and related documents	104
6.11	Documents collected in the “basket”	104
6.12	Entering the title to send collection of documents to EPrints	109
6.13	The Deposit in the EPrints Sword Test Repository	109
6.14	Metadata added with the document	110
6.15	Dropping an item into a repository in the FeedFoward interface (Wilson and Potat (August 2009))	111
7.1	Part of the hierarchy on desktop showing folder documents used in task at level four	116
7.2	Locating a named document at level 2	119
7.3	Locating two associated documents at level 2 given description	120
7.4	Locating a named document at level 3	120
7.5	Locating a named document at level 4	121
7.6	Locating seven associated documents in different directories given description	121
7.7	Comparisons of the Average Times for Windows Visualization and DeFiBro for all tasks	122
7.8	User feedback on the Usefulness of DeFiBro	130
8.1	Presentation of Google results for a search phrase	139

List of Tables

3.1	Documents, files and folder summaries for the 25 Participants	39
4.1	Folder types and their description (Henderson (2005))	59
7.1	Summary of the Files and Hierarchy Levels Involved in User Tasks	118
7.2	Summary of Calculated Times and <i>t</i> -test Results for all Tasks	124
7.3	Precision and recall results for task 1(b)	126
7.4	Time results for task 1(b)	127
7.5	Precision and recall results for task 2	128
7.6	Time results for task 2	129

Acknowledgements

I would like to extend my heartfelt and sincere thanks to a number of people without which this work would have not been possible. First and foremost I would like to thank my supervisors, Dr Leslie Carr and Dr Nicholas Gibbins for their patience, support, guidance and feedback throughout my PhD. Thanks are also due to other members of staff in the IAM and LSL groups especially Sebastien Francois, Dr Gary Wills, Adam Field, Dave Tarrant and other members of the EPrints team who were available to advice, tutor and answer my questions on various aspects of my work.

I would also like to express my gratitude to Dr Steve Hitchcock who made time in his busy schedule to read and comment on the thesis. His invaluable comments enabled me to read and review the thesis from a different viewpoint and gave me confidence in expressing and supporting my work. Thanks also to all the participants who took part in my evaluations and to my bay mates - Melike, Salma and Laila, who provided a pillar of support and good company on the long days spent in the lab.

My PhD would also not have been possible without the support of my family; my mum, brother, two sisters and brother-in-law who gave me unconditional love and support especially during the first two years by taking good care of my daughter while I got settled in Southampton and got my work off the ground. My friends and colleagues back at home, in the UK and around the world were also very helpful and supportive and deserve a very big thank you.

Last but not least I would like to acknowledge and thank the University and Government of Botswana for the generous sponsorships throughout the years of my education.

This thesis is dedicated to my mother, Potlako Mosweunyane, who taught us the value of education and supported us through it all, and to my daughter, Keneilwe B., for the entertainment, smiles, kisses and hugs that gave me a reason to persevere.

Chapter 1

Introduction

1.1 Motivation

Information overload, especially with information in digital form, is a widely recognised and experienced phenomenon (Edmunds and Morris, 2000), as well as being a well-discussed issue in research. Users of computer systems have access to and create and store large amounts of information. Storage has become more affordable (Dong and Halevy, 2005) resulting in an increase in storage capacity of devices, allowing individuals to store more data in what has been termed ‘personal archives’ (Kelly, 2008). Most of this data is in the form of documents in files organized in hierarchies on the user’s computer system. Gemmell et al. (2002) predicted that by now terabyte hard drives, with the capacity to store 2900 1MB documents per day for a year, will be common and inexpensive. This is now the case with even capacity up to 2 TB reported, although desktop hard drives are rarely above 500MB ¹. Users later need to locate, reconcile or extract documents for specific purposes from their document collections. Existing tools for supporting these access tasks require the user to recollect information about the object stored together with its location, therefore placing the burden on the user’s memory (Elsweiler et al., 2005). Bederson and Hollan (1994) also noted that it is interesting that while we can easily process a lot of perceptual data as we experience the world, “we have perceptual access to very little of the information that resides within our computing systems or that is reachable via network connections”. They contend that this information is also, unlike the world around us, rarely presented in ways that reflect either its rich structure or dynamic character.

Motivation for this research resulted from a consideration of the above-mentioned problems and the ideas presented in the field of hypertext’s use in deriving associations between information. Whereas hypertext mostly emphasized the relations between pieces

¹http://en.wikipedia.org/wiki/Hard_disk_drive#cite_note-2TB-15 [last accessed 26/10/2009]

of information within larger structures, a need was envisaged of connecting and associating larger and enclosed items such as documents, adopting and using the principle of hypertext as a way of organizing and accessing information through linking. There have also been major advances in search systems, while there seemed to be a less advances in the provision of exploratory mechanisms which aid retrieval and discovery of information, especially in dealing with personal documents already stored and organized using operating system-provided methods. The author therefore endeavoured to find out methods of improving access to documents.

The Semantic Web ([Berners-Lee, 1998](#)) initiative also brought attractive promises for revolutionizing integration of data on the web, and while it has been applied to associate desktop information in other research efforts, it was interesting to experiment further its utility in solving this particular problem related to documents, and to research any additional benefits provided by structuring the data in this way.

The above-mentioned reasons, coupled with a number of personal frustrations in dealing with documents, served as a motivation for investigating how access, retrieval and discovery in personal hierarchies can be improved. These personal frustrations include always having to remember document locations, traversing long paths to find documents, unsuccessful attempts to formulate appropriate keywords for search and inability to recognise and ‘dig out’ related documents.

1.2 Research Context and Overview

In the past decades information collected by individual users has grown exponentially as storage capabilities increased and disk space cost decreased ([Sweeney, 2001](#)). It has been also been more than two decades since the appearance of the first graphical computer interfaces and operating systems, with newer versions and improvements in operating system functionality produced in-between. Meanwhile the hierarchical file system model has been in use since the introduction of non-visual/graphical operating systems like the CP/M and MS-DOS. The model matches the underlying data storage on devices and was initially used to provide efficient access to files on disk ([Henderson, 2004](#)). Operating system-provided access to information stored using this model is mainly through navigation guided by the structure which is based on the location of the files, and through keyword search.

The user’s desktop serves as a private data-storage and knowledge-creation platform. A large amount of the data is in the form of documents stored in files across folders in the hierarchical file system structure. The system’s view, presentation and support for navigation of this *ad hoc* data storage structure provides a basis for synthesizing and utilizing information in order to effectively carry out knowledge tasks. An investigation

from literature and users shows that support provided by the systems to users is not optimal for productively seeking and working with information in this environment.

The importance of leveraging associative relations between data and information has been recognised as far back as 1945. Visionary Vannevar Bush emphasized the importance of storing and tying items such as books, records and communications together in associative trails such that they can select each other ‘immediately and automatically’ (Bush, 1945). His envisioned personal tool, the *memex*, was to have helped individuals by providing structures to make information accessible instantly and flexibly thereby aiding them in comprehending and remembering relevant information. The *memex* could be consulted through an index or through the associative trail. Provision would also be made for the owner to insert comments of his own and to store his trails for later use and exchange with others.

Several initiatives have been undertaken to help move towards Bush’s vision. His idea was furthered by Theodor Nelson (Nelson, 1965), who invented the concept of *hypertext*, defined as “a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper”. Earlier systems based on this vision such as oN Line System (NLS) (Engelbart, 1962, 1975), Xanadu (Nelson, 1965), KMS (Akscyn et al., 1987), NoteCards (Halasz et al., 1987) and Intermedia (Yankelovich et al., 1988) focused on experimenting with the concept of providing associative *links* between text and between other forms of media, and providing the user with a means to add semantic information to both the materials and links through *annotations*. These formed the basis of research and implementation of hypertext systems in the years that followed. Hypertext evolved into the current web and succeeded in terms of portability and generality, although it lacked the organizational features proposed by Tim Berners-Lee of typed links that enabled automatic analysis (Cailliau and Ashman, 1999).

More recently the Semantic Web initiative (Berners-Lee, 1998; Berners-Lee et al., 2001) was conceived with promises to help in structuring information for easier interpretation by both people and machines, therefore facilitating easier sharing and integration between software components. In the Semantic Web metadata terms and relations between them (ontologies) are defined so that document creators can mark up their documents to enable agents to use them later to perform automated reasoning. The success of the Semantic Web is based on defining and publishing these metadata schemes or formats such that they can be discovered, commonly understood and shared across applications. Several formats have been defined for specific purposes and are becoming widespread across Semantic Web communities. These include the Dublin Core (DC), Resource Description Framework Schema (RDFS), Text Encoding Initiative (TEI) and Metadata Encoding and Transmission Standard (METS). RDFS is an extension of the Resource Description Framework (RDF) that describes application-specific classes and properties. The Dublin core metadata element set, for example, was developed to describe web-based

documents, with elements such as Title, Creator, Subject and Description. The Semantic Web, through access to structured information, enables easier data aggregation (Kobilarov and Dickinson, 2008).

The initiative is taken further by the Semantic Desktop effort, which promotes the use of Semantic Web concepts on personal computers (Semantic Desktop Organization, 2007) with the aim of providing a memex-like personal information system that will “support individuals in their daily activities and augment their memory and intellect” (Oren, 2006). Personal Information Management (PIM) tools such as Haystack (Karger and Jones, 2006), Gnowsis (Sauermann and Schwarz, 2004), SEMEX (Cai et al., 2005), MyLifeBits (Gemmell et al., 2002) and IRIS (Cheyer et al., 2005) are working towards integrating desktop information by applying the above-mentioned concepts, and therefore moving more towards realizing Bush’s vision. These applications are based on integrating information objects on the desktop such as files, emails, and bookmarks and presenting the associations for the user to browse like a personal Semantic Web.

The classification and organization of entities, both physical and abstract, is a fundamental activity of human life. In information science, classification allows for the grouping and arranging of information and/or information sources and provides for their retrieval and dissemination (Walters and Jayakanth, 2001). Advancements in Computer Science have resulted in new and efficient methods of retrieving information through search tools, enabling users to sift through large amounts of information using keywords. However, it has been recognised that the meaning of a classification is highly dependent on individuals, being limited by the fuzziness of cognitive and linguistic boundaries (Humermann, 2006).

On the Semantic Desktop setup the user is encouraged to provide metadata for their documents such that there is no need for categorisation structures like folders hierarchies (Oren, 2006). But the reality is that common operating systems still only provide and require users to use hierarchical file structures for storing documents. In addition despite tools like desktop search engines being available for retrieving with simple keywords, people still categorize into folders and browse them occasionally (Jones et al., 2005). The information categorized in hierarchies needs to be accessed to pursue specific goals, and the process entails integrating existing information and creating new information. To enable integrating information efficient access needs to be supported without asking users to drastically change their usual way of working. In addition users have been found to be reluctant to add metadata to their documents (Kao et al., 2003). Providing and presenting associations between documents on the desktop is thus an important step in working towards better personal information systems.

Figure 1.1 depicts the relationships between the areas discussed and the desktop.

In this thesis we consider the retrieval and discovery of information in the form of documents organized using the hierarchical file system. A combination of the different

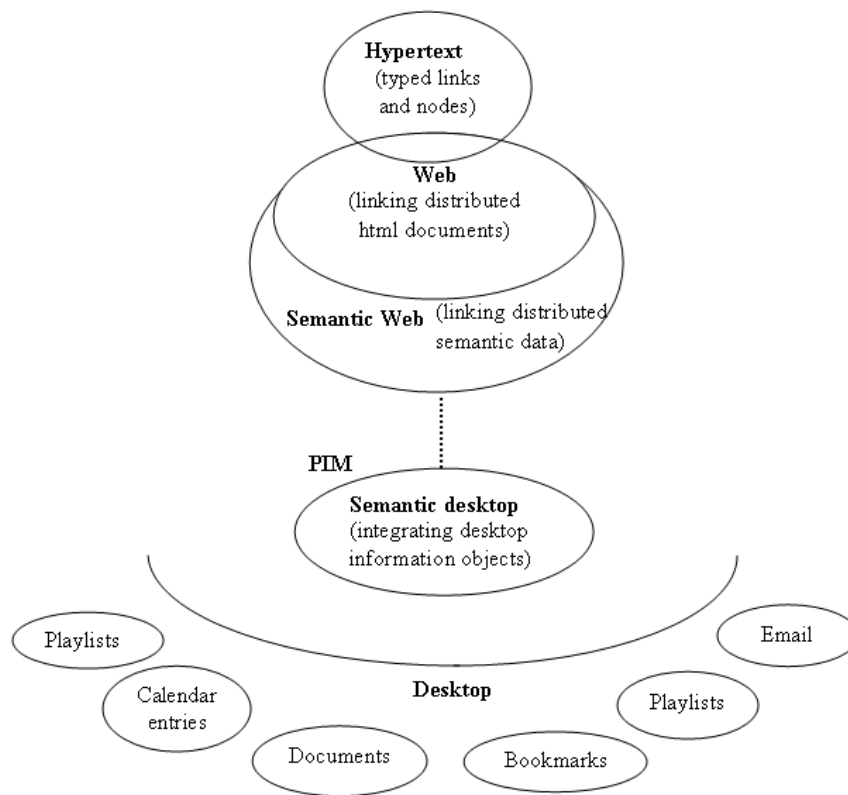


FIGURE 1.1: Data integration approaches feed off data from the desktop including documents

approaches detailed above is employed to propose a solution for improved access and discovery of relations between personal documents through linked and dynamic metadata browsing. Metadata derived from desktop documents is utilized to provide exploratory access methods for retrieving and associating the documents stored in the hierarchical folder structure. The metadata is structured in Semantic Web form to utilize abilities to easily integrate, extend and access the data from different applications. Dynamic links between the metadata terms provide a means to navigate and browse associations between the documents.

The multidimensional interface offered is based on metadata-derived index terms, similarity-based association, and an alphabetic and numeric index, providing a way to navigate the file system and promote visibility and flexibility in retrieving documents. Terms from the hierarchical folder structure forming part of this metadata were specified by the user while building the file structure. These, together with file names and other metadata, are presumed to ease user memory load while also improving precision and recall during document retrieval with the user as they are more likely to match their mental model and vocabulary. Also by exposing these terms users are more likely to discover data hidden in their “personal archives”.

The interface is evaluated against operating system-provided navigation of the file system in tasks involving retrieval and associating of documents across the file hierarchy. The results are presented as evidence that utilization of available document metadata, however minimal, could be useful for helping access hierarchical document stores.

1.3 Research Objectives

The objectives of this research were to:

- investigate people's strategy for storing documents in their file systems and whether they have problems later with accessing and reconciling these documents while carrying out their work. In particular the author had to substantiate the hypothesis that users fail to derive full benefit from their document collections because of limitations of support for re-organizing, recovery and associating documents in their file hierarchies using operating-system provided tools.
- investigate other methods used to support users with working with information, either organized in hierarchies or dealing with documents, and whether and how they can be or have been used to support efficient retrieval, recall and discovery of items in these hierarchies.
- develop a model that improves support to users in retrieving and discovering documents organized in hierarchies in their personal file systems based on information that has proved useful in other methods.
- represent this model through an implementation and substantiate it by comparing it against an operating-system provided method that is commonly being used for document retrieval.

1.4 Contributions

The main contribution made by this thesis is in the form of a model and a proposal for navigation of personal desktop documents hierarchies. The model is based on metadata derived from document properties including hierarchical structure information. It is used to demonstrate a new presentation and access method for desktop documents through an implementation. The implementation is also based on Semantic Web concepts and aspects of similarity, linked and preview-based navigation. The Desktop File Browser (DeFiBro) interface is the platform for this method and is used to demonstrate and evaluate the concepts investigated. DeFiBro's interface presents an in-facet metadata index to offer flatter, more flexible browsing and linked navigation. Support to users is offered in the form of a dynamic browsing structure made up of enhanced facets

and terms which offers user-recognisable filters to browse, select and group desired documents automatically. This interface was hypothesized to support the user to enable them to retrieve desired documents, remind them of documents they may have forgotten about, discover associations between the documents and extract the documents for manipulation.

The interface developed is evaluated by comparing it against the structure-guided navigation offered by traditional systems using retrieval and discovery tasks involving documents in a personal hierarchy. The evaluation is based on measurements of time taken to complete the tasks. The results suggest a statistically significant ($p=0.5$) improvement in retrieval efficiency and effectiveness and exposure of documents in the file hierarchy for better discovery by users, especially when the documents are stored deep in the hierarchy.

The development of this model was informed by a preliminary study and literature review on strategies users employ in organizing and accessing documents stored in the hierarchical folder structures. A study into how users distribute documents in the hierarchy and the extent to which they are "hidden" in these structures was done as part of the analysis of the problem and to help inform the final solution. A set of experiments on specification and use of semantic metadata in filtering and browsing documents was also used to test initial ideas during the development of the solution.

Another contribution of this thesis comes through an important feature in DeFiBro that showcases a method for integrating the desktop with other organized document stores. This provides abilities to easily publish documents with their metadata to these stores. The ability to select and transfer documents with semantic metadata is demonstrated by exporting the data to the University of Southampton School of Electronics and Computer Science institutional repository, ECS EPrints, using the Simple Web Service Offering Repository Deposit (SWORD) protocol. This is presented as evidence that the storage of metadata in a standardized format for all file formats can aid integration and exchange of data with minimal effort on the part of the user.

1.5 Thesis Outline

Chapter 2, *Background*, reviews the ways in which organization and access of documents has been handled by operating systems on desktop computer systems. Current methods of locating and organizing information on the desktop are presented with a view of their shortfalls in helping the user follow connections between information. The chapter also presents a review of propositions for better organization methods based on common methodologies and widely-supported information structuring and visualization methods. Visualization methods applied to file system hierarchies and other methods proposed to replace or improve organization in file system hierarchies are presented and discussed. A

review of methods of managing and integrating personal information based on Semantic Web concepts is also undertaken.

Chapter 3, *A Study of Organization and Access in Personal File Hierarchies*, describes a background study done to find out the extent of concealment of document in the hierarchical file structure, and the problems users have with accessing and recovering documents from this structure.

Chapter 4, *Problem Definition and Solution Design Issues*, refines the problem to be solved. In addition a review of information-seeking behaviour and methods is presented as the beginning of an investigation into formulating the correct solutions to the problems recognised, as well as other design considerations that helped inform the final solution. Metadata are then presented as the basis for solutions to these problems.

Chapter 5, *Metadata Specification and Experiments*, describes the sourcing and specification of documents attributes and presents the experiments done to explore specification, selection and navigation interfaces using semantic documents metadata. A brief review of the Semantic Web is presented as background and justification for its use in this context. Experiments conducted with the document metadata in semantic form are then described, followed by a discussion of the challenges with this approach.

Chapter 6, *DeFiBro: Supporting Retrieval, Associations and Discovery of Desktop Documents*, presents the model and interface proposed for improving retrieval and discovery of documents from the hierarchy that follows a different approach to the one described in Chapter 5. First the model is described, then the implementation work done based on this model follows. The architecture of the system and the interface are described. The chapter ends with a description of a demonstration of a part of DeFiBro that provides the ability to transfer documents packaged with their metadata to other document collections, in particular the institutional repository.

Chapter 7, *Evaluation - Hierarchy Navigation vs Metadata-based Navigation*, discusses the evaluation method carried out with users on the interface developed. User tasks are described in relation to the goal of the evaluation and the real life setup. The evaluation is described in detail and the results are then analysed against the hypothesis and their significance assessed.

Chapter 8, *Concluding Remarks and Further Work*, presents the overall summary of this report and discusses directions that could be taken to further the work that has been done.

Chapter 2

Background

2.1 Introduction

With computer storage devices capacity getting bigger and cheaper individuals have been accumulating more and more data. For personal computer users this data is stored mostly in the form of documents in files on the user's computer system. Operating systems provide navigation methods to access documents stored in the hierarchical file structure, together with advanced search technology and other dynamic methods to help users find, re-organize and group their documents. Researchers have already identified inadequacies in this area and come up with implementations aimed at solving identified problems with organization and access of personal data on the desktop. Some of the approaches taken are based on trusted methods that have been found effective in similar areas while some suggest new approaches of dealing with these issues. These implementations are presented in this chapter.

2.2 Operating System-supported Organization and Access on the Desktop

2.2.1 The Desktop Metaphor and Hierarchical File Systems

Traditional operating systems employ the use of the desktop metaphor for organizing information, where the digital desktop is managed as the physical one, making desktops behave in a more physically realistic manner. The monitor of a computer represents the user's desktop upon which documents and folders containing documents can be placed. A document can be opened into a window, which represents a paper copy of the document. Files can also be spatially arranged on the desktop individually or in groups in different sections of the screen.

A large part of managing documents involves organizing them, and this is done mostly by using hierarchical structures provided by operating systems ([Ravasio et al., 2004](#)). Systems based on the desktop metaphor allow for information to be stored in documents in files that can be named and placed in folders that can be nested to form hierarchical structures. This storage model is used on the most pervasive computer systems such as Microsoft Windows and the Apple Macintosh and is the only work environment known to many users and designers ([Kaptelinin and Czerwinski, 2007](#)).

The desktop metaphor itself has been criticized for being inadequate in terms of handling, presenting and supporting integration of information items ([Kaptelinin and Czerwinski, 2007](#)), and attempts have been made to modify, extend or replace it. Organization into folders involves use of the desktop space, usually for temporary storage before deciding on the right folder to place the document on. Users also reported a mismatch between organizing in a hierarchy on the screen as one can do on the file system, thereby requiring them to put related items in groups on the desktop ([Ravasio et al., 2004](#)).

Current desktop systems provide limited abilities to organize files spatially, temporally and logically ([Henderson, 2004](#)). The hierarchical file system method of organization provides simple and intuitive navigation of the whole file system ([Kaptelinin and Czerwinski, 2007](#)), but it has also proved to be mainly static, and presents problems in categorizing, finding items later and reminding users of what items they have ([Freeman and Gelernter, 2007](#)), among other problems.

Organizing documents into hierarchical folder structures involves categorizing them into specific folders. However, researchers have also found that information does not fall into neat categories ([Lansdale, 1988](#)), but rather falls into overlapping and fuzzy ones which cannot remain unambiguous over time ([Dumais and Landauer, 1983](#)). Malone ([Malone, 1983](#)) and Rekimoto ([Rekimoto, 1999](#)) also identified the problem with users having to classify documents into specific folders because they may belong to more than one category. The categorical structure for files also changes over time and is dependent on the task one might be doing at that time ([Dourish et al., 2000](#)). This means the relatively staticity of the folder system does not make it easy to assign a document to different folders (documents have to appear in one folder at a time), which in turn limits paths by which it can be reached.

[Jones et al. \(2005\)](#) also identify the problems related to use of folder hierarchies. They assert that folders can result in information being forgotten because they obscure information by filing it out of sight. In addition, there can be many folder hierarchies on the desktop supported by different applications for different information, such as email, files and web references. They also reiterate the limitation of presenting information in a hierarchy which imposes ordering while most information has properties which have no inherent ordering and can be used to represent information flexibly.

Classifying documents into folders itself involves a major cognitive task (Malone, 1983), and folders can be viewed as representing conceptual categories in user's file organization (Golemati et al., 2007). Henderson (2005) did an investigation into how users name and structure their folders in hierarchies. She concluded that folder names can be viewed as a user-defined set of attributes or keywords for the document and identified document dimensions (the four most common being genre, task, topic and time) from these which could be treated as facets of document metadata. In addition she recognised that forcing these dimensions into a hierarchy results in duplication across the hierarchy.

On the other hand Malone (1983) studied the way people organized their desks and concluded that computer systems should support automatic classification as much as possible to help relieve users of the cognitive difficulty of categorizing information. He proposed that explicit fields for information like title, author and date be used to automatically classify documents with no efforts on the part of the user. Henderson (2005) also recommended automatic classification based on facets derived from folder names such as person, source, topic, time and file type as these could be deduced from available metadata.

2.2.2 Document Access on the Desktop

2.2.2.1 Navigating the File Hierarchy

Golemati et al. (2007) carried out a study and found that the majority of users still use the Windows Explorer indented list paradigm (or its equivalent in Linux and Mac operating systems) and the simple zoomable visualization that came with Windows-based environments for file browsing despite that there are a lot other visualization methods available. 40% of the test users were Computer Science graduates while the rest were graduates from other disciplines such as Engineering, Arts and Management. This was also observed by Faichney and Gonzalez (2001). On Windows systems, however, Golemati et al. (2007) found out from their survey that most people do not use Windows Explorer, the reasons being that they did not know it exists, they do not find it convenient, they did not need it because they know where their files are and that they never used it and so are not used to it (Golemati et al., 2007).

2.2.2.2 Operating System-provided Search

The explosion of the amount of information available in digital form has resulted in search being one of the most discussed and researched topic in the Information Management Community (Dong and Halevy, 2005), and search engines are striving to provide more sophisticated search technology to improve and further enhance search results. Search engines index distinct terms from documents which are then used to answer queries. But

a complete index only does not guarantee good quality results ([Brin and Page, 1998](#)). As the document collection grows there is need to provide high precision tools to ensure the user benefits from the topmost results presented.

In addition to web search engines, desktop search engines are being offered, with some integrating results from the desktop with those from the web. Newer operating systems like Windows Vista ([Microsoft Corporation, 2006](#)) and the Mac OS X from Apple Macintosh system ([Apple Inc](#)) have incorporated advanced search technologies in their systems that allow users to sort or group results according to their needs. Both operating systems provide powerful search mechanisms based on both file metadata and contents, and methods to dynamically organize the results according to the file attributes.

The operating systems also help users to create personalized views of the search. Tags or keywords added to files help the user and search systems locate more relevant items. For example, in Windows Vista search results can be organised based on file properties like file names, file types, author, or descriptive keywords (or “tags”) that the user have added to their files. The files can also be arranged by type, for example, documents, spreadsheets, or presentations.

The Apple Spotlight search technology provided with Mac OS systems offers advanced search mechanisms based on an index of both content and metadata. Spotlight extracts and indexes metadata about files and other data records including address book records, applications and documents ([Jepson and Rothman, 2005](#)). The indexes are used when a user performs a query either through the Spotlight search window or using the command line. Advanced search options are offered to allow use of names and metadata in sophisticated questions for searches. The search tool also provides the means to search even when only a few details are known, for example, by combining a number of search criteria like “Date:yesterday”, “Kind:Image” or “Kind:Document”. Results from the search can be sorted into categories like documents, images, messages, etc, and further information like previews of images, movies and PDFs and metadata for each file can be viewed. Results can be filtered based on kind, date, people and location. Spotlight can also be configured to control search aspects such as location, date, result grouping, and file types and directories that should be excluded in indexing.

2.2.2.3 Dynamic Organization and Browsing

Dynamic organization is also provided by smart folders in Mac OS X (also known as search folders in Windows Vista) which enable the user to specify a search for items in the user’s hard drive, and save it such that in the future the search could be rerun by just opening the folder. The search will include new items that were saved or created after it was saved. This serves as a more dynamic organisation method which is independent of file location, compared to the usual static folder system.

Mac OS X systems also provide for a user to group or arrange application windows in spaces. Files and folders can be grouped in a stack on the dock at the bottom of the desktop to avoid cluttering the desktop and to give faster access to these. With *Quicklook* one can flip through their documents and previewing is provided without opening the file. The files can also be played in *Quicklook* if applicable. The user can also do this with all the documents they have on their hard drive.

2.3 Access Through File System Hierarchy Interfaces

Hierarchies are one of the most common and important information structures in Computer Science ([Stasko et al., 2000](#)) and have been used in a wide range of applications, from file systems to ontology management and digital library categorizations ([Golemati et al., 2007](#)). Several methods have been applied by researchers to offer improved methods of displaying and visualizing the data stored in hierarchically structured information spaces.

Visualization tools increase the bandwidth of interfaces ([Johnson, 1992](#)) by graphically encoding the information to help humans make sense of and analyze the information set. Hierarchy visualizations have been classified into broad categories such as the indented list (for example Microsoft Windows Explorer), node-link trees (for example two-dimensional node-link diagram), zoomable user interfaces, space-filling techniques (for example TreeMap), and context + focus techniques (for example hyperbolic tree) ([Golemati et al., 2007](#); [Wilson and Bergeron, 1999](#)).

Some of these hierarchy visualizations have been used to depict personal file hierarchies and are discussed below. Researchers have also found some to be more effective for specific browsing tasks and therefore recommended them for these tasks as outlined under each.

2.3.1 Space Filling Techniques

[Stasko et al. \(2000\)](#) compared the use of two tools that implement space-filling techniques (Treemap and Sunburst) using computer file and directory structures. The approaches are mainly convenient for tasks involving file attributes such as type and size.

The Treemap browser depicted three windows as seen in figure 2.1. The *control panel* enabled the user to control the options in the browser, with sets of buttons for changing focus on directories, navigating up directories to their parents, varying depth of file or directories from the root and for controlling color renderings of files based on file age and a random mapping. The *file structure viewing window* depicted the Treemap with directories and files presented as sliced rectangular sections in alternating horizontal and

vertical pieces, with the rectangles corresponding exactly to file size. File types were also colored according to the mapping created in the *legend window*.

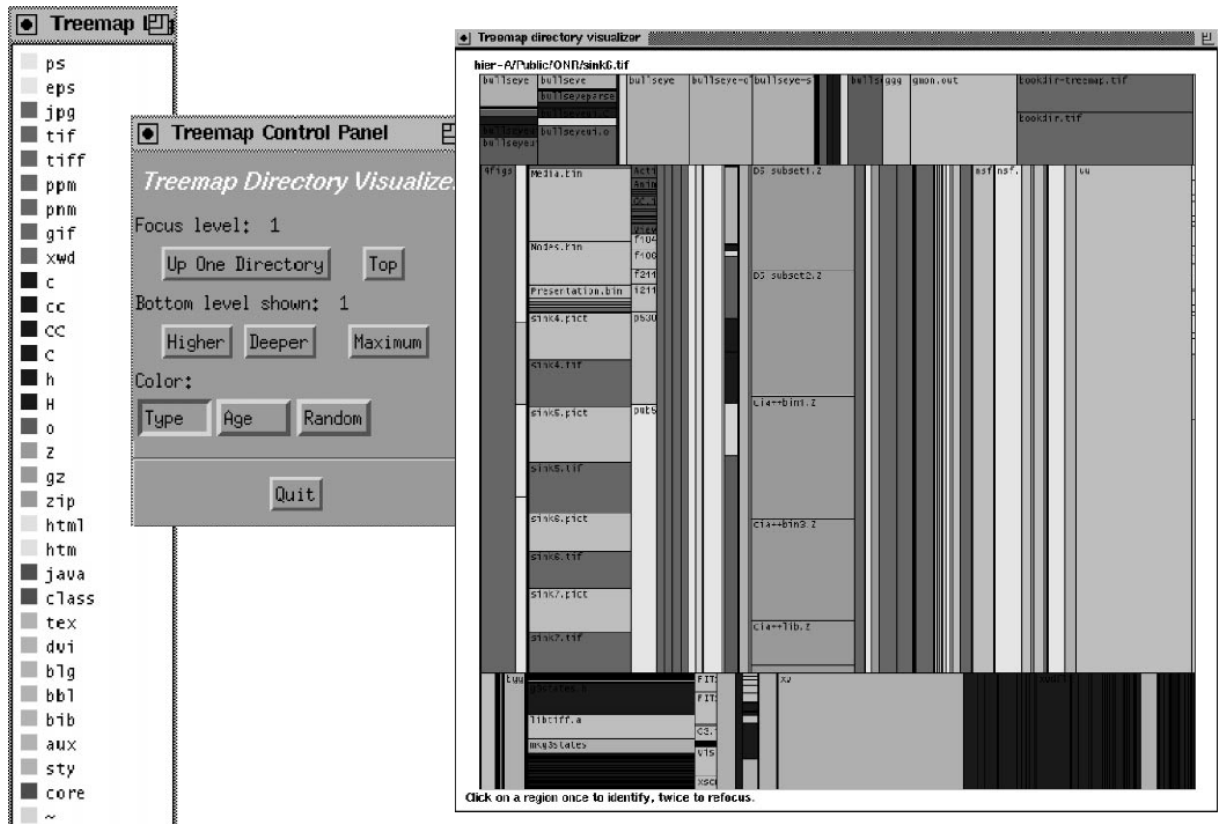


FIGURE 2.1: The legend window, control panel and file structure viewing window of the Treemap browser (Stasko et al. (2000))

The Sunburst browser used the same technique but with the files spread radially with the root directory as the centre and successive directories and files layered as levels from the center. Levels had equal width with the angle occupied by each file or directory corresponding to its size (figure 2.2).

The two browsers performances in assisting users to find particular files and directories or make a comparison between files or directories were evaluated and compared. The authors predicted that the Treemap tool would be better for tasks involving comparison of file sizes while the Sunburst tool would be better for locating specific files in the directory structure. The results showed greater success for the Sunburst tool in terms of learning and depicting the hierarchy structure. While the Treemap tool performed better in tasks involving locating large files and directories, the Sunburst method was seen to perform better for identifying files and directories and dealing with directory-related tasks.

The authors posited that the space-filling tools would be similar to file and directory manipulation tools like Windows Explorer and the UNIX shell in terms of finding particular files but would perform better for attribute-based searches or comparisons.

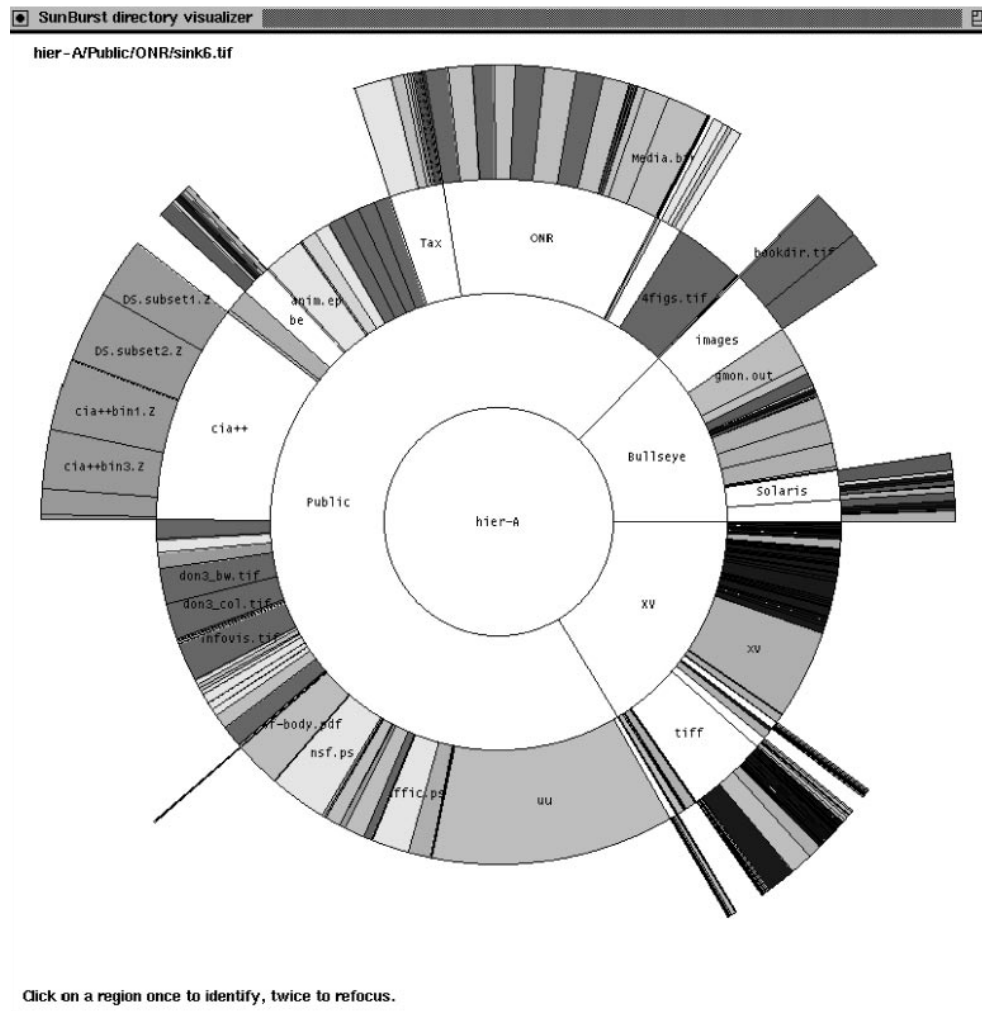


FIGURE 2.2: The Sunburst visualization of a file hierarchy (Stasko et al. (2000))

A similar study was done by Turo and Johnson (1992). The evaluation involved use of browsing tasks to evaluate people use of the UNIX tcsh shell against a Treemap tool. The Treemap tool was again found to be more helpful for tasks that involved differentiating between sizes of files and for locating groups of files with similar attributes. Similarly, the TreeViz for Macintosh (Johnson, 1992) was designed specifically for identifying large files on a disk.

By presenting the hierarchy in a static way, these tools have a limit for the amount of information which can be presented and this is defined by the display space (Turo and Johnson, 1992). When the limit is reached other navigational techniques like scrolling can be used but these may end in loss of context (Beard and Walker, 1990). User disorientation problems involving issues such as offsets, filtering and zooming were reported as a problem.

2.3.2 The Indented List

The outline-oriented view of hierarchic information provided by indented list visualization has been proven to have better performance in most visualization evaluations in which it was used as the baseline ([Golemati et al., 2007](#)). This success is thought to be related to users' familiarity with it.

[Golemati et al. \(2007\)](#) evaluated the use of an indented list tool, Windows Explorer (WE) against the simple zoomable user interface file browser in Windows in tasks involving locating documents and folders in file hierarchies familiar and unfamiliar to the user such that the location of the documents could be known or unknown accordingly. The test users in this context were Windows XP users with at least 3 years of experience using the computer. Contrary to the view that the indented list visualization was familiar to most users, their survey indicated that most people do not use WE for browsing their files, citing reasons including its lack of provision of an effective overview when folders with many children are open and that the users found it confusing. The results of the experiment did not show any statistically significant difference in times taken to locate items in familiar and non-familiar hierarchies using either WE or the simple zoomable file browser. However WE was found to perform slightly better at locating folders and worse for locating files. The problem seemed to be caused by the separation of the folder hierarchy information and the files in different panes, requiring the user to switch their attention back and forth to select the folders and then view the files they contain.

The Windows Explorer user interface has also been criticized for not providing good context techniques, for making poor use of screen space and for not providing enough indication of the document contents ([Faichney and Gonzalez, 2001](#)).

2.3.3 Focus + Context Techniques

The hyperbolic browser ([Lamping et al., 1995](#); [Lamping and Rao, 1996](#)) provides visualization of large hierarchies through use of fisheye distortions to provide focus on a part of the hierarchy without losing context. The hierarchy is laid out on a hyperbolic plane which is then mapped out on a display region. This enables display of the hierarchy in a circle with parallel lines diverging away from each other, allowing the circumference of the circle to grow exponentially in correspondence to the radius. Thus the layout can support hierarchies, which tend to grow with depth.

Selecting a node on the hyperbolic browser results in the display being transformed to bring the current node into focus in the centre of the circle, where initially the root was displayed. Nodes diminish in size as they move outwards. The context is maintained by allocating space for a node based on its distance from the node in focus, allowing a display of several generations of parents and siblings.

An evaluation of the hyperbolic browser testing for effectiveness of practice based on locating nodes against a conventional 2-D scrolling browser with a horizontal tree layout did not show any significant differences. Test users, however, indicated a preference for the hyperbolic browser in terms of getting a sense of the overall hierarchy structure and finding specific nodes by titles. An example of the hyperbolic browser being used to browse a WWW hierarchy in the experiment is shown in figure 2.3.

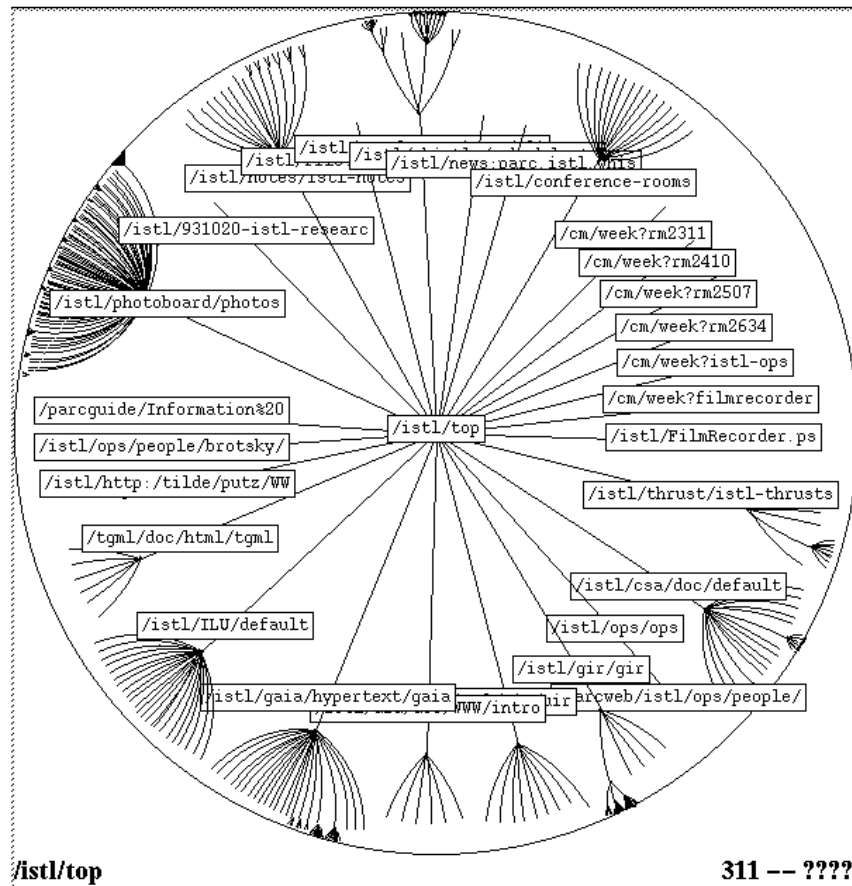


FIGURE 2.3: Browsing a WWW hierarchy using the hyperbolic browser (Lamping et al. (1995))

2.3.4 Node-Link Trees

The SpaceTree browser (Plaisant et al., 2002) is presented as an alternative to the traditional node-link presentation of hierarchies. Node link diagrams are found to make inefficient use of screen space hence requiring multiple screens or scrolling for display. The browser combines the conventional tree layout of trees with zooming that utilizes the available screen space to display the tree branches. Where the available space is not enough to show the branches in full preview icons are used to provide a summary of the branches. The preview icon is in the form of a triangle which is shaded according to the number of nodes in the subtree, with the height representing its depth and the base its width. These dimensions can be chosen to be either relative to the root of the tree or

to the parent of the subtree. The overall orientation of the tree can also be adjusted by the user to match the layout of the natural orientation of data.

The SpaceTree interface also offers search and filtering, allowing for the user to use keywords to find and highlight desired points in the tree or prune the tree to show only the matching nodes. Dynamic queries can also be done to manipulate attribute values to show or hide desired parts of the tree when viewing.

The SpaceTree interface was compared against Windows Explorer (WE), as an example of an expand and contract interface for browsing trees, and a hyperbolic tree browser in navigation tasks involving new and previously visited nodes and in topology evaluation tasks. The hyperbolic tree browser (also known as StarTree) ([Lamping et al., 1995](#)) uses hyperbolic geometry to place nodes around the tree to provide smooth and continuous animation of the tree as the user clicks and drags nodes to readjust the focus point of the tree layout. The interface is criticised for constant redrawing of the tree which can be distracting and displaying labels such that they are not aligned and they sometimes overlap, making them hard to browse.

For finding a node for the first time WE performed significantly better compared to hyperbolic browser in one task (the researchers believe learning might have played a role in this) while for other tasks SpaceTree performed better than WE. Users took advantage of the Hyperbolic and SpaceTree's displaying of multiple nodes at a time to check more than one level at a time, while for WE users showed experience in expanding nodes using labels instead of the icons for expanding the hierarchy views. For revisiting a node, for a long task WE was significantly faster than the other two interfaces, which is thought to have been aided by the ability of WE to have several branches open at once. For short tasks, no statistical significance difference was found between the three systems. For topology tasks both SpaceTree and Hyperbolic browser performed better in two tasks while in the other the Hyperbolic browser was the best performer, but not significantly faster than WE which performed better than SpaceTree. Overall users indicated that they preferred the other two interfaces to WE. SpaceTree was thus found useful for revisiting nodes, making it more appropriate for hierarchies which are used regularly.

2.3.5 Zoomable User Interfaces

Pad++ ([Bederson and Hollan, 1994](#)) is a 2-D zooming graphical interface that was designed as an alternative to the traditional window and icon-based designs. Zooming is used as the main interaction technique instead of menus and pointers. The strategy employed in the design makes use of informational physics to exploit both implicit and explicit semantic relationships and the people's "natural spatial way of thinking" by

supporting different scales of viewing to provide a focal point while also providing context surrounding the detailed area.

As a generic environment for visualizing graphical data Pad++ supports creation and manipulation of multiscale graphical objects, providing semantic filtering of information through different zoom views depending on size or other characteristics. Utilizing hyperlinks the relationship between associated data is graphically represented in the same window and animations are used to center the newly-loaded data and in response to navigation between the links by the user.

The Pad++ directory browser was built to provide a graphical interface for a computer filesystem. In the interface directories are represented by square frames with all their subdirectories and files organized inside them alphabetically. Files are represented by solid squares and are colored to indicate their file type. Filenames and directory names are shown as labels. The user navigates by zooming in and out, or through content-based search. The interface is seen to be effective as a complement to the traditional metaphor-based interface which fully exploits new mechanisms provided by computing systems.

2.3.6 Combination Approaches

The Goldleaf hierarchical document browser ([Faichney and Gonzalez, 2001](#)) was designed to address the problems of Microsoft Windows Explorer related to lack of provision of context, poor use of screen space and poor previews of document contents. The interface employs a combination of two-dimensional, zoomable and space filling techniques to provide a tree-like radial layout of sub folders with documents displayed inside folder windows and represented by large thumbnails. Context is provided by displaying multiple folders simultaneously while filling the screen in two dimensions.

The interface was evaluated against the Windows Explorer user interface in tasks requiring locating documents at different levels of the hierarchy. The results showed at least equivalent efficiency in using Goldleaf in the tasks. A significant difference was noticed in the number of clicks, and therefore the mental effort, required to locate documents in the two interfaces, with Goldleaf requiring significantly less clicks than Explorer, especially for the first three levels of the hierarchy where the folders' labels were clearly visible and when trying to locate a file whose name was not known because the contents could be seen in the thumbnail. This was attributed to the ability of the interface to display multiple levels of the hierarchy simultaneously. Finding higher level folders, however required more clicks as they were smaller in the display and their labels were not always displayed.

2.4 Moving Away from Organization in File Hierarchies

2.4.1 Attribute-based Systems - Placeless Documents

The placeless documents project ([Xerox Parc, 1999](#)) was undertaken as a way of organizing and managing documents using their properties to solve the inflexibility forced by hierarchical organizations which enforce strict categorization. This was done to solve the problem of the inflexibility in organizing and customizing information spaces provided by existing storage and distribution models. The project endeavoured to utilise the concept of the document and its attributes to allow users to interact with information independent of its location. The document properties could be defined by the user according to their needs as anything that has to do with the management of the document, or could be acquired through other means like inference from usage and generation by context analysis. Documents from sources such as the web, email, the file systems, other devices and dynamic processes could be thus be integrated uniformly using these properties.

The Presto Document Management System ([Dourish et al., 1999a,b](#)) was implemented as part of the Placeless documents project. In the Presto system documents can be retrieved, indexed and organized according to their properties, irrespective of their location. The Presto system does not itself store documents, but rather content is stored in external “repositories” like the file system or the World Wide Web (figure 2.4). By storing documents separately from their attributes, uniform management and integration can be achieved across different media. The Presto document object implementation can be run with one or more applications in the same address space, or as a server with Presto applications as clients.

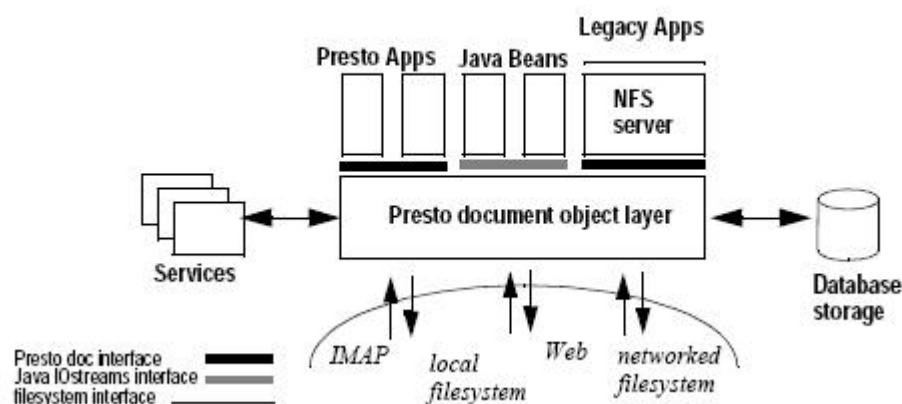


FIGURE 2.4: Presto system core architecture ([Dourish et al. \(1999a\)](#))

The Presto model uses document attributes to provide a uniform means to express document categories, groupings and desired behaviour, as well as to organize and search document repositories. Attributes in Presto are either added by users, especially those

that are more relevant to their tasks, or they are automatically extracted from the system (generic or content-specific). The system uses pieces of code called *services* to extract additional properties for documents by reading, parsing and exploiting the structure and contents of documents. Users can define and add new properties to their own “view” of the document such that the properties are relevant only to them and not the original author of the document. These can also be added to the same document which is in control by other people through personal *references*, which are reference objects to other documents. Documents can be grouped together into collections, which themselves are documents and can be assigned attributes. The properties are then stored as a set of key/value pairs, for example, “Sender=terry@parc.xerox.com”. Pieces of code, called services, are used to extract these attributes at set times or intervals or when particular events occur.

The system provides a query interface and a collections interface which allows persistent grouping of documents which can then be organized dynamically by their attributes and manipulated collectively as a single document, enabling integration across different locations. Fluid collections enable users to specify a dynamic query to select documents, and to do static modification to it through *inclusion* and *exclusion* lists. With these the user can specify documents to include and exclude from the grouping even if they do not match the query.

Whereas the properties are mainly for the benefit of organizing documents by users, they are also available for utilization by appropriate system processes if they are used to specify some coordination between the user and system level. Applications can also utilize the structure to store data associated with the document as an attached property. The Presto object model is also offered to other application interfaces such that custom applications can be developed to exploit the features provided.

2.4.2 Time-based Retrieval

2.4.2.1 Lifestreams

Lifestreams was proposed as a network-centric replacement for the hierarchical directory structure offered by the desktop metaphor. Using a client-server architecture that runs over the Internet, Lifestreams stores a time-ordered *stream* of documents ([Freeman and Gelernter, 1996](#)). Every document sent to or created by the user is stored in the user’s *lifestream*. The user can then create a virtual organization of documents from the stream using *stream filters*. The client-server architecture is machine independent and open such that users can still use applications they are used to.

The basis of creating Lifestreams was arrived at after observing the inadequacies of current software systems and their inflexibility as compared to paper-based systems.

Some of the ideas are that naming files should not be imposed on the user, rather they should choose whether to do so or not, and that categorizing in directories should be done on demand to allow flexibility in storing documents. Other aspects are that archiving and summarization of related documents should be done by the system, and that computers should handle better the task of reminding rather than leaving the burden to the user. The other viewpoint was that access of personal data should be independent of storage devices, allowing universal access of such from data centrally stored on the Internet. These ideas are either demonstrated fully or to some extent in the Lifestreams system.

Lifestream's model allows for users to create or add new, clone, transfer, find and summarise documents into overviews or executive summaries. The system also uses “substreams” to provide the user with a “view” of documents relevant to a search query. Information can thus be organized on demand, and reminders and calendar items can be created in an integrated manner. Lifestreams interface is shown in figure 2.5.

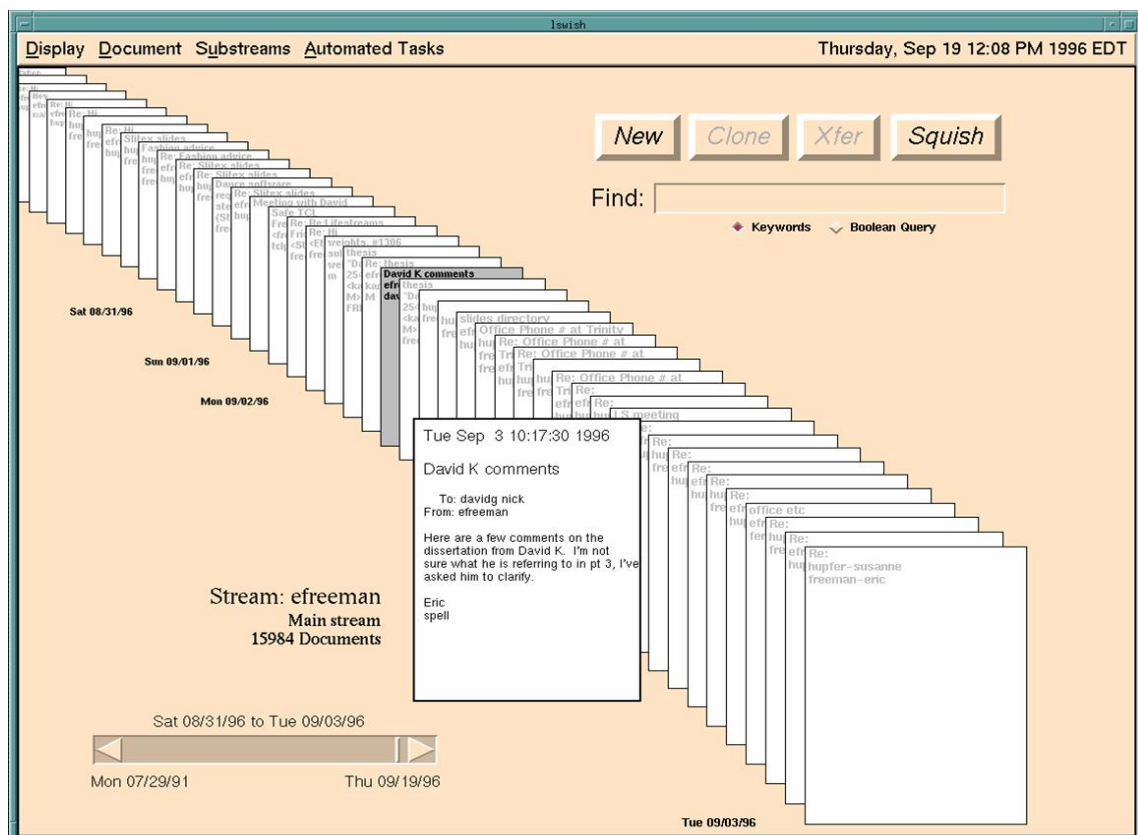


FIGURE 2.5: The Lifestreams interface (Freeman and Gelernter (1996))

2.4.2.2 Spatial Arrangement and Time Ordering - Timespace

Timespace is another metaphor proposed that is based on a combination of spatial arrangement of documents on the desktop and sequential navigation of the documents based on time. The system does not entail the use of folders, all documents are placed

directly on the desktop and can be removed any time. Chronological navigation is allowed to travel to the past or future of the desktop (Rekimoto, 1999), that is, the desktop can be restored to a designated time either backward to a past state or forwards to a future time. Deleted items can thus be retrieved by changing the time of the desktop to the past, or things done for a future time, for example, reminders can be set on a future date. Keyword search is also provided. Figure 2.6 shows a typical Timescape window during user interaction.

The system is implemented in Java and runs on any Java-enabled platform. Attributes such as creation time, deletion time, pathname or URL are maintained for each object on the Timescape desktop. Either the normal file system or a time-aware file system is used. With the normal file system the users see the latest contents of files, even when viewing the past state, while with the time-aware file system both old and new versions are available. A file server is used to record all modification logs onto an internal database.



FIGURE 2.6: Typical screenshot of Timescape (Rekimoto (1999))

2.5 Connecting Information on the Desktop

The tools reviewed in this section provide for integrating information items on the desktop, including connections to reach documents, and the ability for the user to manage and explore the connections between the items. The main aim of these integration tools is to solve the problem caused by fragmentation of information across different applications and devices by bringing it together in one interface.

2.5.1 Semantic Personal Information Management tools

2.5.1.1 SEMEX

SEMEX (SEMAntic EXplorer) is a personal information system offering search-by-association (Dong et al., 2004; Cai et al., 2005). Information browsing is provided through an underlying ontology which can be personalized by users. Users can browse their personal information by semantically meaningful associations previously created to allow for easier later integration by the user. A sample screen shot of SEMEX is shown in figure 2.7.

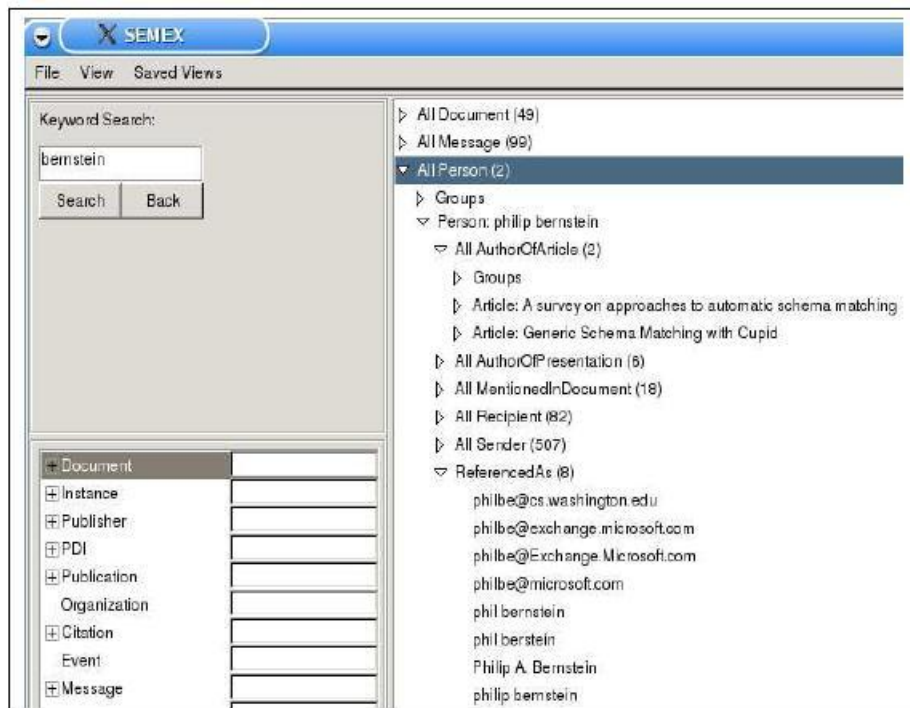


FIGURE 2.7: Sample screenshot of the SEMEX interface (Cai et al. (2005))

SEMEX provides a generic domain model of classes and associations between them and uses this to organize data. This model can also be extended by users, for example, by their browsing pattern. A database of objects and associations between them is represented as RDF, and this is stored and retrieved using Jena. Lucene is used to index object instances by the text in their attribute values. The database supports “on-the-fly” integration of personal and public data by keeping associations and previous activities that the user performed. Users can then browse association links or do keyword search, selection-query search or association-query search. When executing a query, SEMEX also tries to deduce other related objects that are related to the matches found, but not necessarily specified in the query. Heterogeneous data is managed and many different references to the same real-world object are reconciled. The architecture² of the system

²<http://db.cs.washington.edu/semex/semex.html> [last accessed 03/08/09]

is shown in figure 2.8.

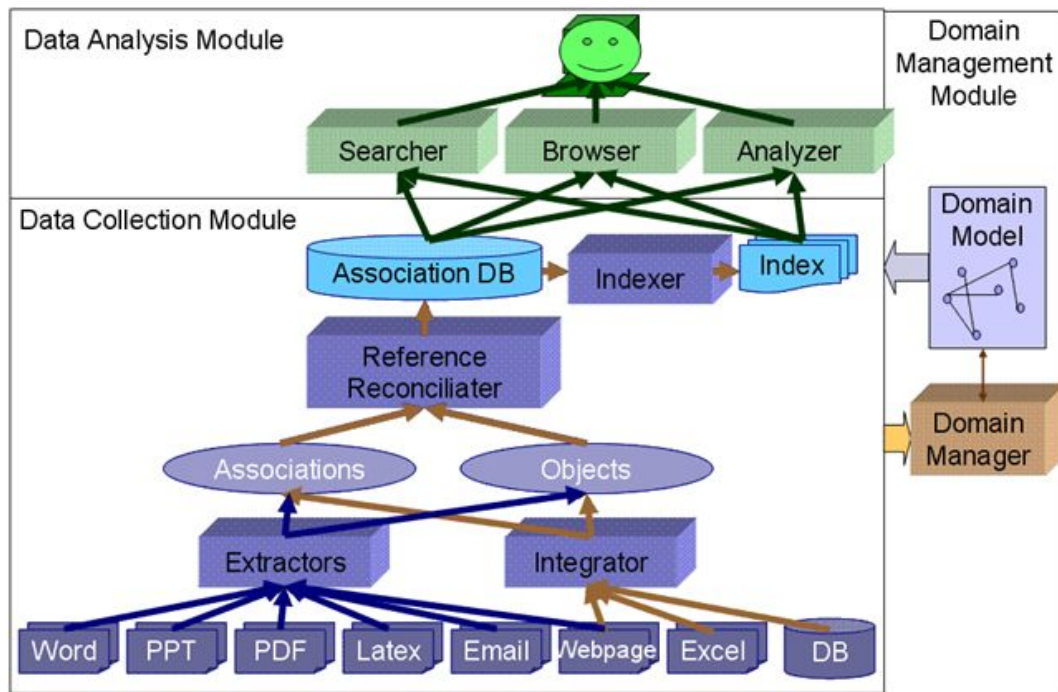


FIGURE 2.8: SEMEX system architecture

The system offers a keyword search, a specific selection search and a facility to browse the information by association. When a query is performed using a keyword SEMEX will return all objects that are somehow associated with that keyword, with the relationship, for example AuthorOf or CitedIn, explicitly specified. In addition those objects that do not contain the keyword but are strongly related to multiple objects in the results are returned.

Search results in SEMEX are ranked by a combination of keyword score, usage score and significance score. The user can also view objects in a chronological order. A sample screen shot of SEMEX is provided in Figure 2.7.

2.5.1.2 Haystack

Haystack (Karger and Jones, 2006; Karger et al., 2003) is a Java-based, open source system that aims to cater for different users' preferences and needs by giving them control and flexibility over storage and retrieval. It caters for storage and manipulation of task-oriented information sources like email, calendar and contacts. Users can define and view connections between their personal data.

A uniform resource identifier (URI) is used to name all individual information objects of interest to the user. These can then be annotated, linked to other objects, viewed and retrieved (Karger et al., 2003). RDF is used to represent the data and to record the

relationships between the objects. The data is extracted from applications and stored in an in-memory database.

Capabilities are provided for users to browse their personal information in one location such that information from different applications and applications such as email, address book, documents hierarchies and the web are brought together in a single view. The user can also add properties to capture any attributes or relationships between the information. These properties can be used as query arguments, for metadata-based browsing, or as relational links to support the associative browsing as in the World Wide Web. A search is also offered as an alternative to the task-specific starting points provided.

Multiple views of the same object are offered to allow the user to use an appropriate view based on their task. Views in the system can also be customised using *view prescriptions*, which are collections of RDF statements describing how a display region should be divided up and which constants and related objects should be shown in the subdivisions. Items could be grouped into collections, and views like calendar view and menu view are provided especially for these. In addition the *lens* view is provided to allow customization of presentation of objects, for example, to show certain properties.

An example of use of haystack to view a user's inbox collection is shown in figure 2.9. The user can view email messages and select people to view data related to them.

2.5.1.3 Gnowsis

The Gnowsis system (Sauermann and Schwarz, 2004; Sauermann et al., 2006) is a semantic desktop³ prototype which aims to integrate desktop applications and the data managed on desktop computers using Semantic Web technology. Desktop resources are treated as Semantic Web resources. A data integration framework is employed to extract information on the fly from common applications. The data and relationships between resources are then represented as RDF. Semantic Web interfaces are added to common desktop applications, allowing the users to browse their desktop like a small personal Semantic Web.

Gnowsis uses a server for RDF data storage, processing, and interaction with native applications and a graphical user interface. To relate information to the user's personal view of the world the Personal Information Model (PIMO) approach is used. The PIMO framework, figure 2.10, is made up of six components. PIMO Basic defines the basic language constructs and the superclass "Thing" of other classes. A domain-independent ontology containing subclasses of Thing is defined in PIMO Upper, while PIMO Mid integrates various domain ontologies and provides classes for Person, Project, Company

³<http://www.semanticdesktop.org/xwiki/bin/view/Main/WebHome> [last accessed 11/08/09]

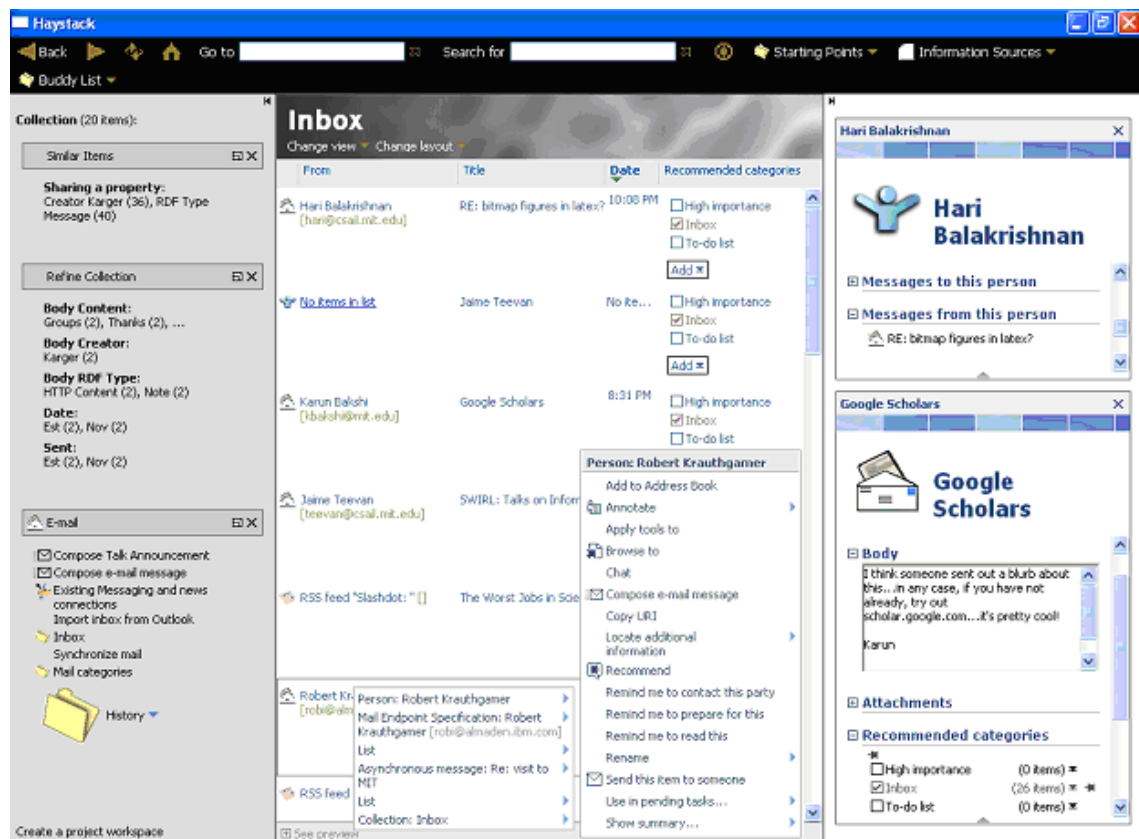


FIGURE 2.9: Haystack showing a view of a user's inbox (Karger and Jones (2006))

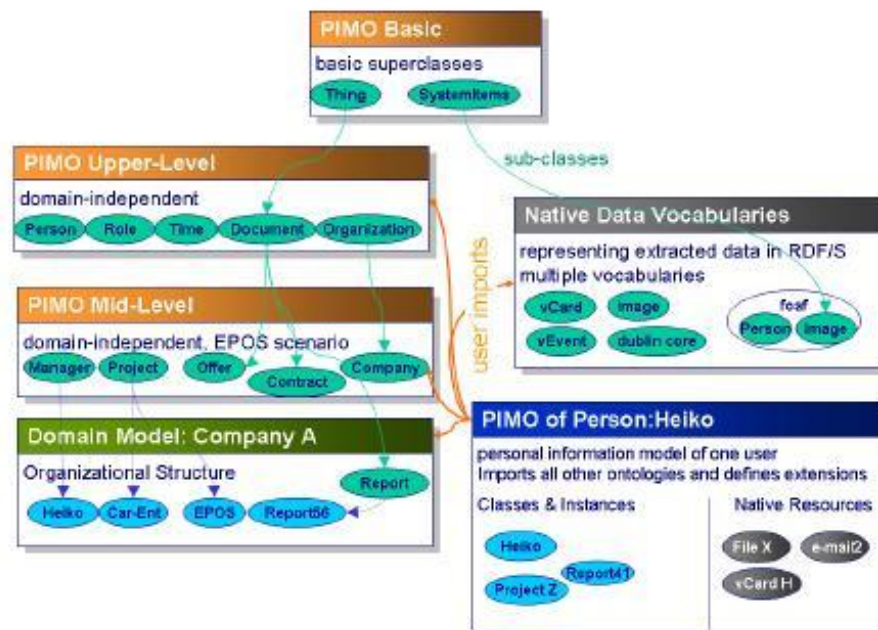


FIGURE 2.10: PIMO ontology components (Sauermann et al. (2006))

etc. The domain model component describes a concrete domain of interest of the user. The user can also extend the above-mentioned models for personal use in PIMO User.

Gnowsiss now tries to incorporate web 2.0 features to the desktop by having users import their tags from tagging websites such as del.icio.us and [flickr](http://flickr.com) and integrate them into their PIMO ontology ([Sauermann et al., 2006](#)).

2.5.1.4 IRIS

IRIS (Integrate. Relate. Infer. Share) ([Cheyer et al., 2005](#)) is implemented as a semantic desktop user interface to CALO (Cognitive Assistant that Learns and Organizes), which is a system that endeavours to serve its user as a personal assistant and help with tasks such as collaborations, organisation of information, and mediating interactions between people. IRIS aims to support users in organising their information resources in ways which suit their individual needs, and can be used on Windows, Macintosh and Linux.

IRIS adopted the Semantic Object Framework, a very fast triplestore implementation, from the Radar Networks Personal Radar (another semantic desktop system). It uses CALO's ontology, the Component Library Specification (CLib), which supports roles, events and axioms that support reasoning. CLib is implemented in the Knowledge Machine (KM), a frame-based language with first-order logic semantics and a reasoning engine written in Lisp. To enable manipulation of hundreds of thousands of data instances required by a user-centric desktop application KM data definitions are translated from CLib to OWL, a W3C-approved standard that supports a flexible data schema, while the axioms are left in CLib. The Jena Framework is used as an API for ontology access because it comes with a set of reasoners, defines a flexible graph architecture and supports RDQL and SPARQL queries. Machine learning is used to create the required links and knowledge that are a semantic representation of the user's work life.

Semantic classes and typed relations are defined and used to integrate data from different applications. These are stored in a knowledge base to provide a persistent store, and query mechanisms are provided across them. Using its plug-in framework applications can embed their own interfaces within IRIS. This three-layered architecture is depicted in figure [2.11](#).

The “embedded suite” of applications are hosted such that semantic events are captured as they occur and real-time suggestions are offered as the user works with information. Users can create a “personal map” across their office-related information objects. The project also aims to build collaboration infrastructure in the future.

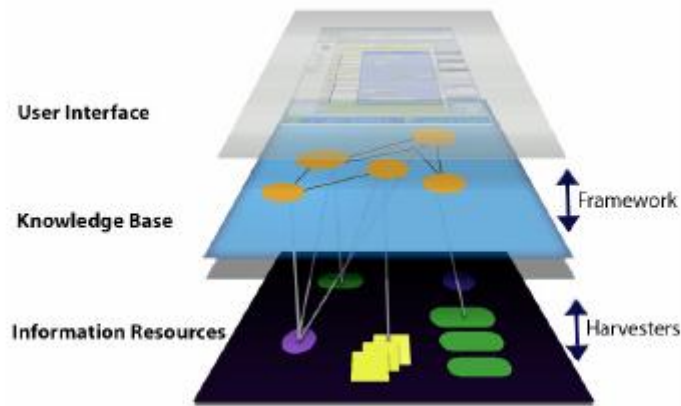


FIGURE 2.11: IRIS framework showing the three-layer architecture (Cheyer et al. (2005))

2.5.2 Database-based Methods

2.5.2.1 Iolite

Iolite (Intelligent On-Line Inferencing for Text and Email) (Rothrock et al., 2006), another information management tool, provides a unified interface for navigating associations between information in different file formats and applications. The system uses agents to discover associations by observing the user and monitoring activities on the computer. This is done through either direct observation of associations or inference through machine learning and other intelligence. The system also allows manual construction of the associations. The associations are then used by the Iolite clients to display associations for selected items. The system represents associations as an undirected graph stored in a relational database.

The system can also be used to replace the standard dialogs for opening and saving files or as a stand-alone application like the Windows Explorer.

2.5.2.2 MyLifeBits

The MyLifeBits project is an effort that works towards achieving Bush's vision of the Memex (Gemmell et al., 2002, 2006). It is based on the user keeping all their digital artifacts in a personal storage repository and not being restricted to a hierarchy to organise it. Non-text media can be annotated and transclusion (two-way) links can be created by authoring tools. It uses an SQL server database to store resources, their metadata and links, and full-text search is provided with the help of Microsoft's Index Server. Links are a resource's annotation of another resource, and a resource can annotate or be annotated by any number of resources.

Many visualisations are essential to help users understand their “life bits”. The MyLifeBits Shell allows query results to be viewed in detail (list of resources and their property), thumbnail (miniature images), timeline (thumbnails on a linear time scale set to hours, days, week or months) or clustered time (thumbnails clustered by similar time and arranged in time order) modes. Resources can be annotated easily with text or audio, and audio is changed to text to allow a search to be performed. Resources can also be assigned to one or more collections.

Figure 2.12 shows the architecture of the system.

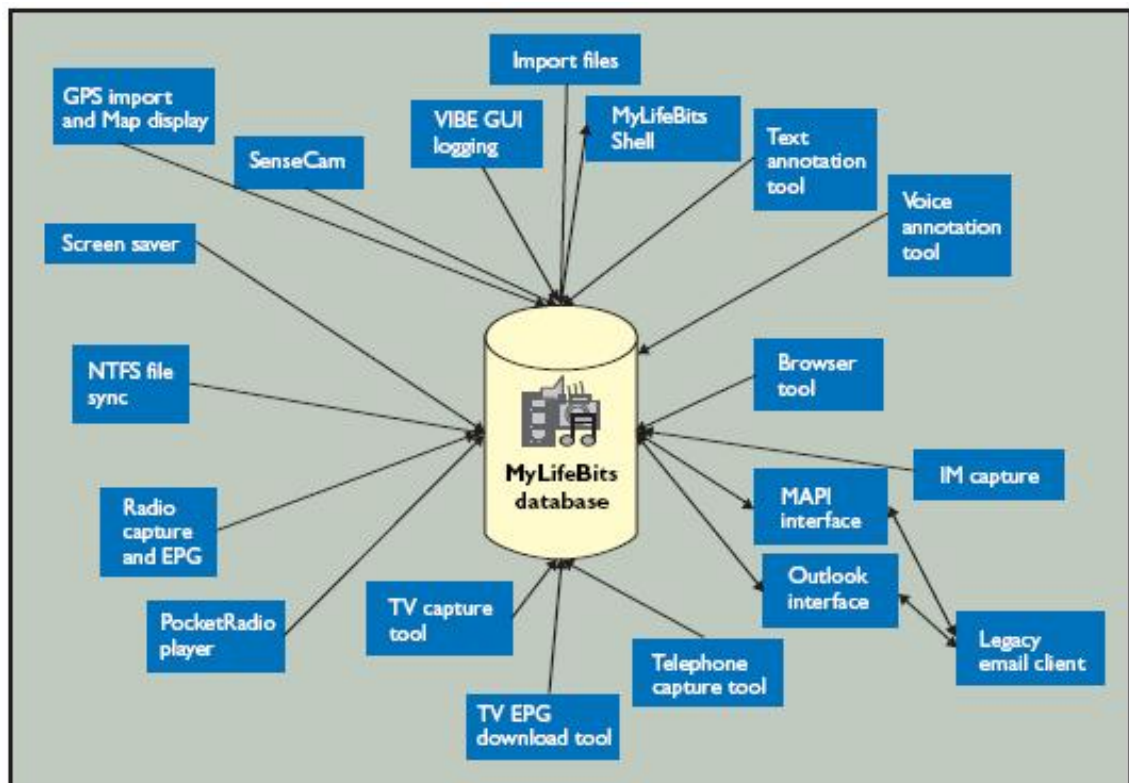


FIGURE 2.12: The MyLifeBits platform store and the suite of capture/display tools (Gemmell et al. (2006))

2.6 Discussion

2.6.1 Problems with Organization and Access on the Desktop

The organization of documents in hierarchical folder structures and the inflexibility of such a structure in supporting categorization, recovery and association of documents has been widely reported in literature. The inadequacy of operating system-provided methods for access and navigation of this structure has also been discussed. The proliferation of search tools and dynamic organization methods that come with the new operating systems only solve the problem to a certain extent, failing when the location

of a document or associated documents is not known or when the user finds it difficult to remember and specify information related to the document.

Hierarchical browsing has been reported to facilitate information seeking in tasks in which the information is not known or is not specific ([Walters and Jayakanth, 2001](#)). File system hierarchies, on the other hand, are usually personal implying that the user is familiar with the information and occasionally looks for specific information contained therein. The efforts reported under hierarchy visualizations to improve the presentation of file system hierarchies and their comparison against system-provided navigation in most cases suggest a general view that there are limitations with methods provided to users to view and manipulate file system hierarchies in current systems.

Evidence from the efforts and implementations discussed in this chapter also show that integration and associating data is a topic of major concern in desktop computer systems, and that computer technology still has to solve the problem of finding and reminding ([Malone, 1983](#)). Space, time and attributes are the main organising factors in the systems described.

2.6.2 Current Solutions and Influence on Proposed Solution

2.6.2.1 Hierarchy Visualizations

The suggestions and attempts for improvement of visualizing hierarchies emphasized the need for provision of overviews over a collection, together with striking a good balance between provision of context and detail. Hierarchy visualization methods seemed to do well at providing context and overview by utilizing the available screen space for displaying the whole structure. Expand and contract or zoom mechanisms are used in some cases to change focus according to user needs while still maintaining context. Space-filling techniques and node-link trees in particular have proved more suitable for tasks involving grouping or comparisons between files based on attributes. These techniques have already been found to perform better when users are more concerned mostly about leaf nodes and their attributes and do not need to focus on the topology of the tree, or the topology is trivial ([Plaisant et al., 2002](#)). Good provision of overview and context through these visualisations therefore aids comparisons more than locating and navigation or viewing connections between information items.

For locating documents, provision of both context and detail such as content has proved helpful. The indented list interface which is provided by most operating systems for navigating the file hierarchy has been criticized by users for poor provision of context and for locating files. Solutions such as the Goldleaf browser strive to provide both context and detail through a combination of approaches. Whereas provision of context was found to be useful for locating the first items, details however, seemed to be more compromised as more items are viewed on the display, making them difficult to locate.

However, this and the other solutions presented above, while providing good models for retrieval and recognition of groupings between documents based on some attribute, still use hierarchical representation, utilizing the available screen space to either show the parts being navigated, or to present it in its entirety. Visualization of context is highly dependent on and may be limited by the amount of information in the hierarchy (for example, in the user study conducted in Chapter 3, users had an average of 39102 files in 2839 folders). Even utilizing the whole available screen space might therefore not be enough to help the user make sense of the collection.

2.6.2.2 Attribute-, Time-based and Spatial Systems

These systems are based on or are an attempt to extend or replace the desktop metaphor provided by standard operating systems. The desktop metaphor itself has been viewed to be inadequate and has even brought problems as well as solutions ([Ravasio et al., 2004](#)). Presto uses user-defined and system-derived document properties to enable dynamic and static organization through collections and queries. Lifestreams and Timescape employ time and spatial arrangement combined with time respectively to enable navigation and retrieval of documents. With these the organizing strategies provide useful views of documents that could help in their retrieval.

2.6.2.3 Personal Information Integration Tools

Personal information management tools aim to integrate data items on the desktop to enable the user to traverse information through associations. Through the use of Semantic Web technologies or databases, the data items, their properties and associations between them are encoded and used to present users with an interface for dynamically browsing the information associatively. These tools therefore provide ways of locating documents through connections with other pieces of information on the desktop.

2.7 Summary

The tools discussed in this chapter support different visualizations and browsing interfaces to support management and retrieval of desktop information based on property-based groupings or classifications. Annotation support and search are also central to most of these systems. For those which employ Semantic Web concepts an ontology is defined to capture the various concepts, things and events to be represented and explored. While some reuse elements from the widely-distributed metadata schemas, some systems use their own elements in their ontologies, or a mixture of both. Knowledge bases, databases or triple stores are used to store the RDF data and relationships.

Indexing of attribute values for search is also commonly supported. Various programming languages are used, although most use Java, while a few employ logic, reasoning mechanisms and machine learning for data access and integration.

The various solutions reviewed were aimed at the problems of classification and access of desktop documents and have revealed a need to investigate the problem further in relation to locating and linking related documents. An interface offering a different approach to presentation of the information in the hierarchy and connections between them might be more desirable in aiding retrieval, association and discovery of documents stored in file system hierarchies. Attributes have proved useful in all the implementations reviewed, therefore utilizing attributes, both user-defined and system-derived, might be beneficial to the approach.

Recommendations for creation of tools for organizing, effectively viewing and retrieving information stored in the form of documents include

- **Automatic classification** - Some of the critics of systems' requirements for classification using the hierarchical folder structure recommend automatic classification using attributes. The intention is to relieve users of cognitive difficulties in classifying. But if users do not classify their documents it might be difficult to support integration according to user's needs since they would not have provided essential attributes which could be used to derive relations between the data themselves. Some of the systems reviewed such Presto ([Dourish et al., 2000](#)) recommend provision of attributes by users which are then used to manage documents. Users, on the other hand, have been found to be reluctant to add these to documents ([Kao et al., 2003](#)). Other systems reviewed in this chapter do utilize document attributes either as a basis for virtual organization (timeline tools) or for depicting the files in the hierarchy for comparison purposes (hierarchy visualizations).
- **Exploration** - Users of desktop systems have problems accessing their document structures. Most information in documents remain unutilized because it is "hidden" in these data structures. People have been seen to prefer browsing to search, but not adequately supported in doing it. Categorization is effective in aiding retrieval and the categorization structures (folders) can be utilized to allow dynamic reorganization.
- **Dynamic re-organization using attributes** - Users are also been encouraged to define and add metadata in the form of attributes and keywords to accompany their documents. Meanwhile apart from use in search systems, there is little reward for putting an effort into this in current systems in terms of reorganizing their document spaces.

Researchers like [Rothrock et al. \(2006\)](#) and [Henderson \(2004, 2005\)](#), have emphasized the importance of organising one's documents well, to avoid spending considerable time

locating information in the future. While this is desirable, it is not possible to achieve perfect organization on the desktop using the current operating system tools. Since methods for automatic classification are not yet available on current operating systems, a method has to be devised to assist users in locating and integrating information by exploiting information already available to dynamically restructure and view documents and their relations on the desktop.

Chapter 3

A Study of Organization and Access in Personal File Hierarchies

3.1 Introduction

A preliminary study involving file hierarchies on personal desktops and problems users have in seeking information from these structures was undertaken in order to understand user needs and form a basis for problem-solving in this project. First, data was collected on a selection of users' desktops detailing the structure of their file hierarchy in terms of the number of documents (Portable Document Format, Microsoft Word, Excel, Powerpoint presentations, and htm/html), files, folders and subfolders contained therein and the depth (folder level) of the files within this structure. The aim of this data collection was to find out the approach taken by desktop users in storing documents in terms of how they are nested in the folder hierarchy. Secondly, a questionnaire was disseminated to the same users whose data was collected to find out about their strategy and problems in organizing and accessing documents in their hierarchies, as well as their viewpoint on the support given by the operating system in these respects.

The research had already posited, through evidence from previous research, that most users encounter problems with effectively accessing and utilizing their personal documents organized in the hierarchical folder structures supported by the current systems. The background study was therefore carried out in order to verify and further clarify this view.

3.2 User Group

Participants for study were randomly picked from a heterogeneous group of regular computer users in the University of Southampton. All the participants had had enough experience using computers, more than 3 years of regular (almost everyday) use. The group also formed a broad spectrum of Microsoft Windows, Linux and Apple Macintosh (Mac) desktop and laptop systems users who regularly deal with documents on their computer systems. A total of 25 participants were recruited for this study. 16 of these were Windows users, 6 used Linux while 3 were Mac users. Of these 4 were undergraduate students, 11 were postgraduates, 9 researchers and academics, and one was a software engineer.

The participants came from different backgrounds; 12 had a background in Computer Science, 8 in Computer Engineering, with other individuals coming from Civil Engineering, IT, Telecommunications Engineering, Physics and Computer Science, and Business Administration (Marketing). The age ranges were as follows: 18-25 (7), 26-33 (13), 34-41 (3) and 42+ (1). The group consisted primarily of Computer Science and Engineering students and researchers mainly because there was a need to ensure that participants were familiar and comfortable with the terms used to describe features on desktop systems and the notion of hierarchies. This might however make the results biased towards these particular types of users and the file numbers might indicate a certain type of use for the files. For example, a lot of their files might be those related to programs implemented or downloaded to the desktop as is the usually done by this type of users. Future work has to therefore consider the diversity of test participants and endeavour to represent as many groups as possible.

The participants were required to sign a form to agree that they they have given informed consent to take part in the study and were free to withdraw at any time. Before the study was conducted an application was made to, reviewed and approved by the School's Ethics Committee to make sure that risks related to the identity of the participant, their consent and data gathering and use were addressed.

3.3 User Hierarchy Enumeration Study

This part of the study did not involve any active participation by the user. Scripts were used to automatically collect the information from the users' desktops without any intervention. Though the user was not necessarily needed their presence was essential for a couple of purposes. Firstly, the scripts had to be run in full view of the user and both the script and the data collected were available for inspection to allay any concerns the user might have had about their privacy being invaded or their personal data being taken away. Secondly, the user was needed to provide information on where

their document hierarchy began, that is the top level folder, such that an initial point (disk drive or folder) can be determined to provide a starting point for the data collection. For example, in Windows systems most users created their folder hierarchies straight from the desktop, which is considered the top of the hierarchy, requiring the start point to be

`<drivename>:/documents and settings<username>/desktop`

while in Linux and Mac systems most users had a dedicated data disk drive to store all their data, requiring the start point to be

`<drivename>.`

Some of the Linux and Mac users had their personal files spanning several partitions, requiring the start point to be the root directory. The script was thus ‘instructed’ to collect data from the points identified as appropriate by copying it to those locations and by running it from there.

Two different scripts doing the same thing had to be devised. A Visual Basic script was used to collect the data from Windows systems while a Perl script was used on Linux and Mac systems. Although the Perl script could be used on Windows systems this also required a Perl interpreter to be installed. On the other hand Perl comes packaged in Linux and Mac systems while VBScript is included in Windows systems. Based on these facts and that most Windows users do not have this component installed and that installation by the investigator would be impractical and imposing on the users, a decision was made to use the appropriate script for each of the systems.

The script collected data about the folder hierarchy and the number of files contained therein. The data collected consisted of the number of top-level folders, number of sub folders, number of files and the depth of each file within the hierarchy, that is, in how deeply nested within the sub folders it is from the top-level folder. This gives an indication of the amount of documents kept by users and where in the hierarchy they are kept. These data were collected without recording any user or file information. The collected data was also anonymized and stored confidentially and will not be kept longer than necessary for analysis and presentation in this thesis.

The document types surveyed were limited to those commonly used, that is, office documents (Portable Document Format, Microsoft Word, Excel, Powerpoint presentations, and Hyper Text Markup Language). The aim was to try to get a vision as accurate as possible of documents created and organized by the user as opposed to other files on the system that were stored by applications, for example during installation.

3.4 Qualitative Study on Working with Documents on the Desktop

A questionnaire was disseminated to the same users after the data collection from their desktops. The questionnaire was used to obtain the users' perceptions regarding the use of metadata and to answer questions on whether and how the current tools provided by the operating systems gave support for locating, collecting, extracting and working with documents. The questions specifically asked about

- *Addition and use of metadata*- whether the subjects utilized the documents' properties feature for addition and editing document attributes as provided by desktop systems and whether they needed to reorganize views on their document collections using these attributes.
- *Browsing strategies and views on documents collections* - how the subjects get to view their collections to get a sense of what is available.
- *Users' views on how organization, associating and transferring documents on the desktop could be improved* - this is based on their view of how the operating systems support tasks involving working with documents across the hierarchical structure.

The questionnaire used is included as part of this report in Appendix [A](#).

3.5 Results and Discussion

3.5.1 User Hierarchy Enumeration Results

The user hierarchy enumeration study revealed a number of issues which may have to be considered when dealing with user file system hierarchies and these are discussed below.

1. Number of documents, files (total) and folders.

First, the Participants were found to have a varied number of documents, the range being 23 to 16 698. The mean number of documents in this sample was 4575, with a median of 1435 and a standard deviation of 5715, indicating a very wide spread of document numbers. Similarly there was a wide range and spread in the total number of files stored and the number of folders they are stored in (table [3.1](#)). An informal discussion with the participants revealed that most owned more than one machine, in most cases a desktop and a laptop used in different locations (home and office). While an overlap in the sets of files kept in each machine was reported,

	Mean	Median	Standard Deviation	Range
Documents	4575	1435	5715	23 - 16698
All Files	39102	9165	54704	121 - 187109
Folders	2839	1307	4200	13 - 15664

TABLE 3.1: Documents, files and folder summaries for the 25 Participants

it seems almost all the work files were kept in the office, hence the ones covered by this survey.

It is also interesting that while the number of folders seemed to increase with the number of documents, a substantial increase in the number of files did not result in a corresponding increase in the number of folders (figure 3.1).

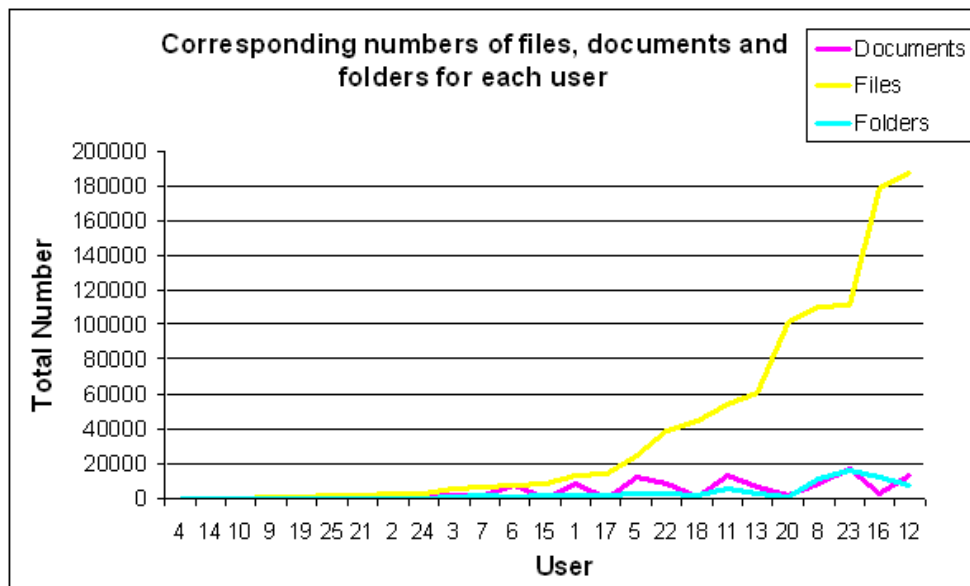


FIGURE 3.1: Relationship between numbers of documents and files with folders they are stored in

2. Organization in folders for documents.

The number of subfolders the documents were nested in ranged from 0 (documents on the desktop or top-level directory) to 23. Figure 3.2 shows the average number of documents stored at each level, that is, inside that number of subfolders, 0 being at the top-level directory (desktop). From the chart it is clear that most of the documents were stored inside less than 12 subfolders (97%). Further analysis showed that 93% of all the documents were stored in less than 10 folders, that is, they were stored in 0 to 9 subfolders. More than half (56%) of these were stored in 0 to 4 subfolders while the rest were nested in 5 to 9 subfolders. These findings are

not consistent with findings by Golemati et al. (2007) that most users rarely use more than 4 levels to store files. The participants are similar in this case to those of Golemati et al. (2007) (around 40% of both test users are Computer Science graduates with the rest from other disciplines from a university setup). However further investigation is needed to find out why the documents ended up stored at those levels, that is, if the users themselves created levels over time consciously or they were created through other means, for example while creating backups.

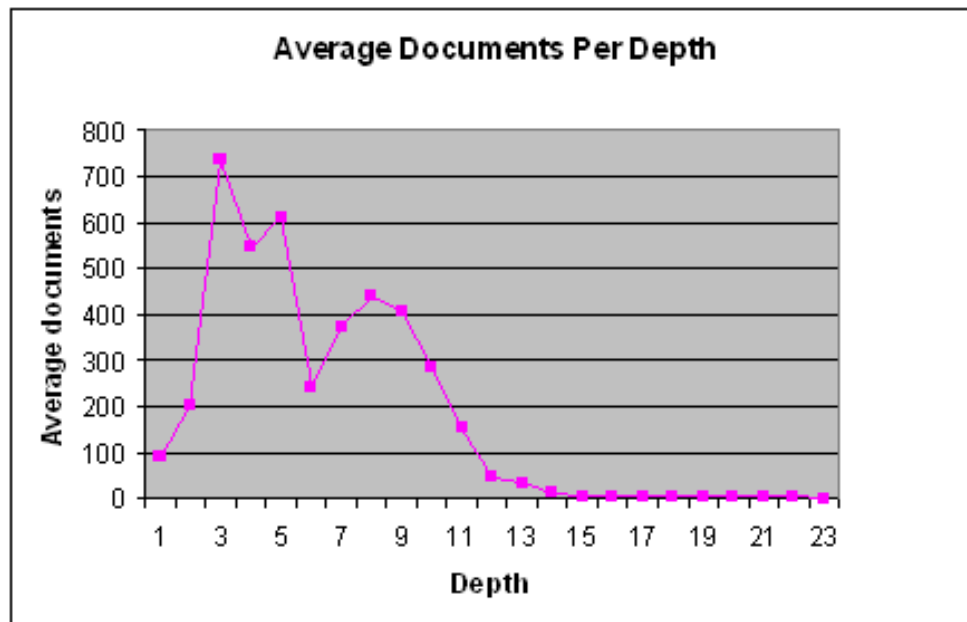


FIGURE 3.2: Average documents stored in the number of levels

3. Levels used versus number of documents.

In terms of the user hierarchy storage structures 44% of the users used deep hierarchies (more than 8 levels) while the rest, 56%, used shallow ones (up to 8 levels) (see figure 3.3). About a third of the users used 10 levels or more while at the lower level the same percentage used 6 or less levels.

The general pattern revealed by the results was that users with fewer documents (less than 1000) used fewer levels (mostly 5 or less). Figure 3.4 shows the total number of documents each user had against the total number of levels the documents were stored in, depicting a general increase in the number of levels as the total number of documents increased.

3.5.2 Questionnaire Results

The answers given by the participants to the questions given revealed the following.

1. On the use of Metadata for Documents.

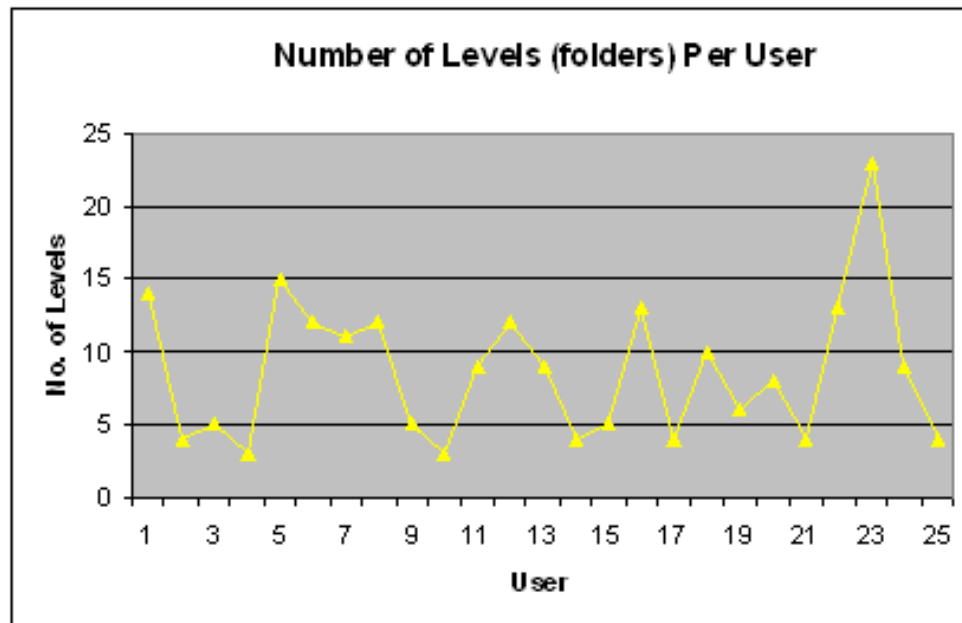


FIGURE 3.3: Number of levels used by each user

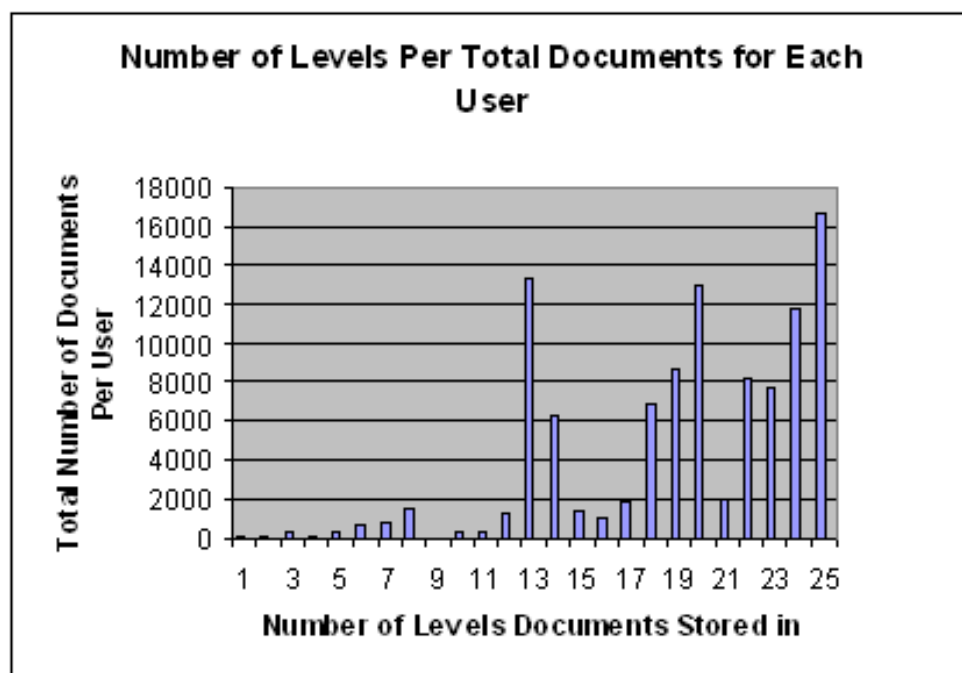


FIGURE 3.4: Number of levels used by each user having the given number of documents

- About half of the users reported that they rarely or never edited inbuilt document properties while the other half did sometimes or often. But 80% of the users reported using information provided by the document properties for locating correct files while browsing, for re-organizing, for search and also viewing them while working with or inspecting the documents. This was the view irrespective of the operating system used.

2. On organizing and accessing documents using system-provided methods.

- More than half of the users reported needing some kind of overall view of their document collection. Almost all Microsoft Windows users said that according to their knowledge, this feature was not provided by the operating system. Two admitted not knowing and one user was not sure if it was provided or not. Some Windows users reported utilizing other available functionality like advanced searches and Windows Explorer “detail” view to achieve a desired view. Some users reported resorting to manual tactics like writing down file names as one goes across folders to derive desired document groupings from across the hierarchy. A few reported employing third-party file administrator tools to achieve this.
- Most Linux and Mac users however, reported that they utilized a variety of tools by the system to deal with their document collections (Mac’s Spotlight search tool, recursive listings on Linux, Finder on Mac to find documents), with two Linux users reporting having no problems at all when looking for or working with documents across the hierarchy. However these users, and even these two, indicated that there is need for improve the system for browsing documents with features like previews and allowing addition of custom metadata.

Users’ skills play an important role in how users do things. A considerable percentage of computer users are low- and medium-skilled users who use computers extensively but, compared to the power users, do not get adequate benefit from expending effort in the use of the computer given their limited knowledge ([Ravasio and Tschertter, 2007](#)). Most of these low- and medium-skilled people use Microsoft Windows, while fewer people, who are mostly power users, use Mac and Linux systems. One can therefore still conclude that functionality provided by all systems for accessing documents from today’s systems is not optimal.

- Users with a high number of documents especially reported problems and needing help with “extended searching and filtering”, finding documents stored deep in the hierarchy, getting a quick view of documents’ contents, dealing with redundancy and recognizing and working with related documents across several folders.

- Users at the other lower end of the spectrum, that is those with fewer documents, still reported having trouble with finding, reconciling and working with documents across several folders.
- It should be noted that none of the Windows users stated 100% satisfaction with organizing and accessing their files and this might be interpreted to indicate that there is still a gap to be filled regarding this.

3. On improvements needed for documents' organization and access.

- Browsing, grouping and views - participants expressed needs for improvement of presentation and visualization of the file hierarchy structure to enable better contextualization of both files and folders within the structure, viewing by desired groupings, and provisions of previews to enable quick checks while browsing. Example of statements used to express these respectively:
"Good if pointing to a folder the user can see the metadata as well as the hierarchy tree of the subdirectories"
"I think it will be useful if the user can choose several folders to be shown in a tree format of their subdirectories and files"
"I find browsing through pdfs painful. Previews on Vista are very helpful. We are also providing previews, as much as possible, in EdShare (edshare.soton.ac.uk)"
- Collecting together desired documents for use or transfer elsewhere - participants indicated difficulty with selecting files from across the file hierarchy and transferring such between folders or to other media. Advanced users reported adopting techniques such as batch processing on Macintosh OS X while naive users adopted simple means like opening several folders at once, searching several times and aggregating the results and even manual means like visual inspection and documentation using a word processor.
- Linking and Reminders - Participants expressed a desire for the system to aid them in recognition of related documents and provision for browsing the logical links between them. "Remembering which is the master (latest) I want to keep. Not always sure where they are being saved to when use saveAs."

3.6 Summary

The main aim of the hierarchy enumeration data collection was to find out how users store documents in the hierarchical file systems in terms of the number of documents stored and how deep in the hierarchy the documents were stored. The user feedback on organizing, accessing and working with documents on the other hand provided information on user needs. The results can be summarized as follows.

1. Users accumulate personal documents which are scattered across, and sometimes embedded very deep in their personal folder hierarchies. The pattern followed seems to be that the more files or documents a user has, the deeper the hierarchy.
2. The classifying of documents in a hierarchy only serves the function of organizing for re-finding and reminding to a certain extent. Users reported inadequacy of operating system tools in providing different dynamic views and previews, visualizing relations or links between documents and inadequate support for working with or creating desired groups of documents. Users try to utilize available tools provided by operating systems to achieve desired results, often resorting to third party tools.
3. There is also evidence of manual work and use of external applications to supplement the shortcomings of current operating system in working with documents. The shortcomings can be summarized as lack of presentation and visualization mechanisms that allow selection, grouping, recognition and browsing of links between documents as well as both contextualization and detailing of documents and folders on the desktop.

The results confirmed the hypothesis that most users encounter problems with effectively accessing and utilizing their personal documents organized in the hierarchical folder structures supported by the current systems.

The next step is determining how users can be supported in organizing and working with their personal documents given the views expressed and the available computing support structures, which is essential for productively synthesizing ideas and effectively carrying out their tasks.

Chapter 4

Problem Definition and Solution Design Issues

4.1 Introduction

The importance of organizing and integrating information for retrieval and manipulation on the desktop has been emphasized in the previous chapters. While today's search systems are considered advanced and sophisticated enough to handle the process of locating documents simply based on information such as filename, location or contents of the document, there is still need for systems that improve dynamic organizing, integrating and presentation of such information on the desktop as demonstrated by research efforts in this area. In this chapter it is argued that the problem of finding and relating is still not solved as we still need systems that connect information. Users can still benefit from organization systems that help group or integrate information to help them in case they cannot remember what they are looking for, or to help them discover information "hidden" in their document collections.

This chapter therefore refines the problems to be solved and discusses possible solutions to these problems, including why document metadata can form the basis of these solutions. A review of the process, methods and design features that warrant consideration when dealing with information seeking tasks is also undertaken as part of looking into considerations that should be taken for the final solution.

4.2 Shortfalls in Aiding Retrieval and Association of Desktop Documents

4.2.1 Finding and Reminding on the Desktop

Documents stored in file system hierarchies can be reached by navigating the structure, using location in the hierarchy as a clue (Gonçalves and Jorge, 2004). When navigation based on location fails as a retrieval mechanism, search engines are usually used. Search however requires the user to remember some information about the document and to specify it as search terms. Hundreds of results can be returned in the form of a list, requiring the user to sift through the results again to find what they are looking for. In addition the *vocabulary problem* (Furnas et al., 1987) which is caused by users using the “wrong” words and therefore failing to get the information they want, may further aggravate the problem. Keywords assigned by indexers often do not match those tried by users in searches, resulting in low recall rates (Furnas et al., 1987).

In comparison, access to other digital resources such as those from the web or in institutional repositories can benefit from the use of Knowledge Organization Systems (KOS), such as ontologies and thesauri. In these domains vocabulary control and knowledge organization (Tudhope and Nielsen, 2006) are used to reduce ambiguity of terms used to describe and retrieve resources by providing structures such as context and synonyms that aid resolution of meaning of the terms.

Golemati et al. (2007) concluded based on their study that most users avoid very deep hierarchies for storing their files, rarely using more than 4 levels because they find it difficult to browse deeper hierarchies later to retrieve documents. From the wide range of user’s surveyed they also found that most users, even the most organized ones, had trouble refinding a file or folder when it is requested suddenly and out of context. Suggestion for improving navigation included visualization of parent and sub-folders without having to open them, showing previews on “mouse over” and offering an alternative file system that would make multiple categorization or file tagging possible.

While users have created rich knowledge structures in the form of file and folder hierarchies current systems offer little to present this knowledge back to remind the user of the knowledge already specified.

4.2.2 Relating Documents

Organising documents in folders does not help much in finding *related* documents later. For example, if the folder hierarchy represents tasks undertaken, or is based on time or place, documents created by one author may be in different folders depending on the basis for their arrangement. Similarly, finding similar files across folders based on

attributes such as date, application, other metadata fields added by the user such as subject, category and keywords is not easy on desktop systems.

This problem has also been recognised in locating related documents across different repositories (Swierk et al., 2002). With the availability of different personal storage devices like laptops, desktop computers, hand-held devices and web storage devices, most of which require organization in hierarchical folder structures, the problem of locating and managing similar documents is intensified.

The major problem that must be solved is therefore that of associatively connecting information, such that the associative connections can be followed by browsing and users can be reminded of the information they have which has been organized into a knowledge structure.

4.2.3 Proposed Solutions: Browsing and Exploration Interfaces

It has already been established by many researchers that users prefer to find information by orienteering and browsing between related items rather than jumping to it directly by search (Teevan et al., 2004). Search has also been considered inadequate for dealing with personal information (Gonçalves and Jorge, 2008). Users have been shown to be reluctant to use search tools, using them only when they could not think of where the documents were stored (Ravasio et al., 2004). Rather, they spend more time traversing the directory structure (Dong and Halevy, 2005). Barreau and Nardi (1995) noted that users preferred browsing in order to scan and recognise rather than try to remember the filename or use search facilities. They suggest that this is probably because location-based filing engages more actively the mind and body and imparts a sense of control. Orienteering also appears to decrease users' cognitive burdens during searches, giving them more control by maintaining a sense of where they were, and giving them a context of their search results such that they ended up with a better understanding of the results (Teevan et al., 2004).

People have also been found to prefer exploration not only to discover information in an unfamiliar domain (schraefel et al., 2006), but also when investigating their own personal information space (Cheyer et al., 2005). Based on this view Oren (2006) reasoned that since we often find it difficult to unambiguously specify what we are looking for, our memory can greatly benefit from context information which can be enabled by exploratory browsing.

In addition, search engines can be of limited use. For a website, it is difficult, for example, to search for a document written by a particular person, to search within a particular category of the web site or to combine a variety of search criteria (Kelly, 2000). This also applies to desktop search. Studies have also shown that users prefer navigating or "orienteering" (steering towards the desired information in a series of iterative steps)

instead of using search tools to jump directly to the information (Rothrock et al., 2006; Ravasio et al., 2004).

4.3 Design Lessons for Information Seeking Interfaces

4.3.1 Information Seeking and Information Retrieval

Information seeking is “a process in which humans purposefully engage to change their state of knowledge” (Marchionini, 1995). It involves interaction with texts involving seeking texts and active construction of meaning from those texts. Using computer systems, this may take the form of search or browsing and is often accompanied by uncertainty (Bates, 2007). Unlike the limited notion of the term “information retrieval”, Marchionini (1995) views information seeking as a more general, dynamic, higher level cognitive process (as part of learning and problem solving) that does not necessarily imply that the information was known and organized by the user or someone else in advance. This makes information seeking more closer to question-answering and knowledge acquisition than information retrieval. The process is also seen as a broad, pervasive aspect of human behaviour which requires a user-centred, interdisciplinary approach to fully understand (Rice et al., 2001).

Even so, Belkin (Belkin, 1993) proposes that information retrieval be considered as information-seeking behaviour which involves more than just a process on selecting texts based on some static specifiable need, but active interaction of the user with the texts and the user having control, as opposed to the system, in the activities, such as comparison and representation, associated with the process. This view therefore requires support of the user as the central component with the system as the intermediary and both having responsibilities and control over the process, identification and support of information-seeking behaviours in specified situations, and support of the process of interaction with texts as being central to information retrieval. This can be done by controlling the interaction, representation corresponding to the user’s interpretation of the interaction and comparison use to suggest answers during interaction (Belkin, 1993).

Information seeking, according to Marchionini (1995), depends on the interactions of the person seeking the information, the setting, task, knowledge domains, system and outcomes. The seeker has a *mental model* of the system, “a cognitive representation of a problem situation or system that is active in the sense that it can take inputs from the external world and return predictions of effects for those inputs”. It can thus be “run” internally and the results used to make decisions about actions. Mental models are determined by the conceptual models provided by the designer through the interface and allow us to both understand problem situations and predict consequences of actions contemplated for solving problems. Users develop mental models for systems through

reading documentation, training, experience with systems, and comparing them with previously encountered systems (Marchionini and Shneiderman, 1988). The feedback from the system typically alters the user's knowledge and influences how they conceptualize the problem and undertake the task, and how subsequent search activities are conducted (Rice et al., 2001). While the user's cognitive processes that guide information seeking are not known and understood, mental models can serve as a guide for building and testing the theory of information seeking (Marchionini and Shneiderman, 1988).

4.3.2 Information-seeking Strategies

Five information-seeking strategies have been identified by Canter et al. (1985):

1. Searching - involves the user seeking a specific target based on a particular motivation. (This is also known as *directed searching* (Bates, 2002), to distinguish it from the general term "search" and is done after the information items have been indexed or categorized by some means first).
2. Browsing - the user follows the data wherever they are led until they satisfy their interest.
3. Scanning - the users covers a wide area shallowly without much attention to detail.
4. Exploring - users survey the extent and nature of the field by following many different paths, possibly with influence from a motivation to find a particular target.
5. Wandering - navigation by the user in an unstructured way, with possibility of revisiting some data.

Strategies for search can be divided into *analytical* strategies and *browsing* strategies (Marchionini, 1995). Analytical strategies involve careful planning, formulation and reformulation of query terms together with examination of results. Browsing strategies, on the other hand, require less cognitive load but are more heuristic and opportunistic and depend on the user's attention and recognition of relevant information. Browsing is also viewed as a part of human behaviour that takes place in various contexts of daily life because of its tendency to require less effort (Rice et al., 2001) and it has increasingly assumed greater importance especially in human-machine interaction (Croft and Thompson, 1987; Oddy, 1977).

Browsing involves a cycle consisting of a series of glimpses followed by closer visual or manual examination of selected targets, which may or may not be acquired either physically or conceptually, before moving on to the next target (Bates, 2002, 2007). The

process ignores any kind of organization of the items being examined, though the glancing takes advantage of the associations in the arrangement to locate the most appropriate area to be searched, followed by a scan of relevant materials in the area identified (Bates, 2002). Rice et al. (2001) view browsing as consisting of four dimensions; a *behaviour* (characterized by scanning - looking, examining or sampling where the eyes and body move at one's will), *motivation* (presence of an intention even in presence or absence of purpose and goal), *cognition* (non-specification, minimal or extensive definition of objective) and *resource* (form and user's knowledge of the resource). The process involves a user scanning a resource, and they end up pursuing other interests that they did not consciously have and were not actively searching for but end up pursuing them as a result of stimulated association with other items surrounding them. This concept, known as *associativity* and defined as making associations or linking information stimuli, is seen as a characteristic of the browsing activity in seeking objects based on a search goal.

Marchionini (1995) identifies three general types of browsing based on the information needs of the seeker and the approach taken. *Directed browsing* is systematic and focused, based on a specific and known information need and requires repetitive browsing of an object. In *semidirected or predictive browsing* the information need is not as clearly defined and the approach is less systematic, and requires multiple attempts to solve the problem. In *undirected or general browsing* there is no defined goal and very little focus is required.

Linking which involves the *pursuit* of connections made between parts of a document or between documents, is also seen as a type of information seeking technique by Bates (2002). The advantage of links is that they are aimed at specific targets, but they may also be liable to unexpected changes and may also be incomplete, resulting in the user ranging widely but without major accomplishment across the relevant materials (Bates, 2002).

4.3.3 User Actions in Information Seeking

Kwasnik (1992) identifies six behavioural activities that play a role in browsing:

1. *Orienteering to the environment* a continual activity, that happens not only one at the beginning, but is developed and modified during the activity to get an idea of the structure and content of the dataset. The activity has also been otherwise known as *directed situated navigation*, and involves navigating in a small series of steps using contextual knowledge as a guide (Teevan et al., 2004).
2. *Place marking* - A view can be marked for potential subsequent consideration. This may either be physical or mental and is liable to change as purposes and interests are reshaped by other encounters during the process.

3. Identification - Involves a recognition of items that may be interesting or are identified as definitely not interesting. This depends on the person's ability to condense and comprehend the view contents presented.
4. Resolution of anomalies - posed by the structure and content of the environment, which the browsers themselves have tried to resolve through dynamic structure creation and orientation during browsing.
5. Comparison - between items, environments and features in the environment. These serve to guide, identify and confirm purposes and aims.
6. Transitions - Movements from view to view. Transitions may be movement toward an object that is expected to help fulfil a goal, or away from one, when something has been identified before and rejected or existing information has been exhausted.

[Schneiderman \(1996\)](#) recommended support of information actions that users wish to perform while seeking information through visualizations. The actions are presented as seven tasks; Overview, Zoom, Filter, Details-on-demand, relate, History and Extract to allow for rapid information presentation and user-controlled exploration through the interface. Although these were recommended for use in designing graphical user interfaces they summarize overall helpful aspects in information interfaces and their application in the design of other information seeking interfaces may help in supporting the user through interfaces dealing with presentation of information collections. The tasks are summarized below.

- Overview - information visualization interfaces should support some overview strategy to help gain an overview of the entire collection.
- Zoom - focus on a portion of the collection.
- Filter - focus on interests and eliminate unwanted items, e.g. using dynamic queries.
- Details on demand - select an item or group and get details when needed, e.g. select item to get values of each of the attributes.
- Relate - view relationships among items
- History - keep a history of actions for undo, replay or further refinement
- Extract - allow for extraction of items for other uses.

4.3.4 Browsing and Search on the Desktop

Browsing differs from search in that users can immediately experience the information space without formulating an initial query (Kules and Shneiderman, 2003). Barreau and Nardi (1995) found out in their study of how users organize their files on the desktop that users prefer location-based finding to logical finding. Location-based finding involves the user taking a guess at a directory/folder where they think a file might be located, going to that location and then browsing the list of files or array of icons in the location until they find the file they are looking for. Logical finding, on the other hand, involves conducting a text-based search to locate the file. Browsing is thus considered the best option for organising and locating associative connections between documents.

It is however important to consider search in relation to browsing. From the system's perspective browsing is essentially a search with no defined starting point (that is, the system is trying to satisfy the request but has not been provided with a start point in the form of a query). In addition, to group items for browsing, a query (essentially also a search) is done in the background. Search also provides a convenient starting point in a browsing system, hence some of the systems reviewed include some search mechanism.

Browsing is an activity that users have been undertaking to look for documents, hence document collections should be organized for browsing to give users an overview of what is available and how different items are related to each other (Henderson, 2005). Enabling users to associatively browse their document collections without the formulation of complex queries has been also been advocated for by researchers for a long time but has not been effectively implemented such that it can be widely adopted on desktop systems.

An index can serve as an important aid when the user does not have or cannot clearly express their information need (Bruza, 1990). In such cases an index can help clarify this need by presenting the information instead of requiring the user to formulate a query.

Bruza and van der Weide (1990) define *associative navigation* as navigation in which the user follows cross references that uni-directionally thread connected information together, this being characterized by bringing about a change in context.

4.3.5 Facets For Organization

Facets are aspects, properties or characteristics of a class or subject which are clearly defined, mutually exclusive and collectively exhaustive (Taylor, 1992). With regard to organizing collections, facets partition collections along orthogonal sets of categories (Yee et al., 2003). Facets can be either flat or hierarchical, and can also be either single-valued (one value to an item) or multi-valued (multiple values to an item). They allow an item to appear simultaneously in multiple taxonomies (Yee et al., 2003).

Faceted filtering or clustering can help users get an overview of collections (Kobilarov and Dickinson, 2008). Users are presented with facets which they can use to filter information by selecting facets, then values within the facets to get smaller sets from the collection. Systems that provide faceted browsing have been based on metadata, for example (Yee et al., 2003), with some like mspace (schraefel et al., 2006) calculating the facets automatically. Although facets can provide a quick overview by clustering data collections along certain dimensions their usability can be greatly reduced as data sets grow with potential facet values also increasing (Kobilarov and Dickinson, 2008).

Koren et al. (2008) classify facets into five types. *Nominal* facet types are characterized by discrete and orderless values, with the values occurring only once per facet (e.g. authors). An *ordinal* facet also has discrete values, but with each value having an implicit ranking (e.g. letter grades). *Interval* facet types have continuous and ordered values with no absolute zero point (e.g. copyright year), while *ratio* types are similar but contain a true zero (e.g. page count). *Free* types are special types with facet values made up of multiple, possibly repeating tokens (e.g. synopsis).

The process of facet development begins by defining the subject to be covered by looking at existing classifications or thesauri, or titles or subjects in the database. The terms defined are then organized or broken down into facets with the relationships between them specified (Aitchison et al., 2000). The items are then organized under these facets which differ from the main group by that characteristic. For documents on the desktop the existing classifications are folders created in the file hierarchy by the user. Further classifications can however be derived from the document attributes in those fields that are common amongst all the documents, for example author, organization, file-save-time. Although these facets can exhibit a hierarchical relationship, further analysis is needed to determine and provide these facets. For now the documents can be organized by the categories already identified, which excludes the folder hierarchy since it has been ‘flattened’ in our case.

4.3.6 Document Clustering

Clustering techniques aim at categorizing documents based on some measure of similarity (Hearst, 2006). The measure is usually based on some document representation information such as index terms or vectors which are seen to represent best the meaning of the document (Marchionini, 1995). The documents are classified by automatically assigning them to a predefined set of categories (Käki, 2005). These methods have been applied in search and retrieval systems as a way to group similar documents that tend to be relevant to the same queries. This has a potential of improving recall as queries are matched against clusters and the documents in those clusters returned as results, but have been criticized for being too slow for a large corpus of documents and for not significantly improving retrieval (Cutting et al., 1992).

Rasmussen (1992) classifies clustering algorithms into two types: hierarchical, which results in a multiple-level categorical structure of the items, and non-hierarchical (also known as partitional (Cutting et al., 1992)), which results in items being partitioned in a one-level categorical structure with mutually exclusive groups. Hierarchical clustering techniques include single-, complete- and group-average linkage.

Other methods used to categorize and relate documents include term extraction, which can be undertaken manually or automatically. Using the *Kea* algorithm (Jones and Paynter, 2002) keyphrases are extracted automatically to act as document surrogates and are used to augment document metadata, to refine search query results and to generate links inside documents or present them for browsing as metadata and to show related documents.

4.3.7 Recommendations for Design of Personal Information Systems

Oren (2006) discusses 6 issues that affect the design of personal information systems that reflect on users' needs for augmentation of memory and intellect.

1. A lack of good and thorough activity models of the knowledge worker - including the need for tool support in document management bearing in mind workers' high personal autonomy and freedom for creative work.
2. A misperception of the mental model behind categorising - users' mental model for categorising is unknown. People remember associatively and retrieval clues have to relate to the same context as the categorised information to stimulate these associations.
3. An underutilisation of the interlinked nature of the information - support for navigation should be scalable, support graph-based navigation and not dependent on a fixed schema and allow exploration without prior knowledge of its structure (exploratory interface). Annotations must be exploited for more precise searches and links for exploratory navigation. There should be provision for automatic presentation of related information.
4. A lack of appreciation to the context of activities - the context of information is important for recall. Information is classified, recalled and remembered along the facets what, who, when, where, and why.
5. An unawareness of the use of physical paper documents - recognition that reading and writing in paper is still important because of its suitability for quick annotation and stimulation of reading, and should be integrated into the electronic world.
6. A tendency to be overambitious in the suggested level of support - there is no need to strive for systems that "understand" information as only the user can give

meaning to documents. Support should therefore be focused at capturing user information and making it easily accessible to the user.

Based on these Oren therefore recommends focusing on individual users first, leaving users to their freedom and not constraining them to rigid schemas, exploiting the interlinked nature of information, remembering context and using it to enhance recall and understanding, valuing the paper and keeping it simple by capturing and representing things user wants to store before reasoning.

4.3.8 Discussion

The surveyed approaches provide further insight into how users look for information, how they interact with information seeking systems, and how they can be supported when embarking in the process. Search, browsing and following links are the methods mostly used to seek information through interfaces. Search involves the user specifying what they are looking for, while browsing and link-following enable exploration without any precise specification of the user's needs. Through their mental models of the systems, users interact with and are influenced by the systems through the results provided, their knowledge of the domain, the environment in which they are doing work and the tasks they undertake during problem solving.

Document representation and organization methods are used to serve as basis for browsing and search. Such methods include representing documents using indexes, vectors and keyphrases followed by clustering and categorization based on some technique for determining similarity.

Browsing, in particular, is advocated for as an effective mechanism in aiding retrieval and discovery of information items in a collection. Informed browsing, enabled through provision of a broad range of clusters and contextualization of information is especially desirable for scanning and exploring information spaces. Overviews of a collection, which can be provided through facets, indexes or clusters, can give the user a sense of what is available in the entire collection and to assess, clarify or reformulate their need. Systems based on strict hierarchical categorization are increasingly recognised as being poor for effective exploration and browsing of information collections ([Hearst, 2006](#)) and as such flexible, dynamic browsing support is viewed as desirable. An exploratory interface that utilizes the linked nature of information and filters information upon user requests to provide associative navigation seems attractive for this option.

Understanding the user's needs and actions during browsing is essential for designing interfaces that provide support for user interactions through appropriate presentation of information and aspects in the interface. Notable issues include provision of overviews (through either facets or spatial visualization or some form of clustering) and details,

ways to mark views and compare items, extraction facilities to select and isolate desired items for action on later and history facilities.

In this project the goal of browsing is for retrieval of a either a known or unknown document given some information need, or a need to discover information items and related resources. Therefore browsing that utilizes the information already specified and available to support informed, linked navigation based on some form of clustering is desirable.

4.4 Document Attributes as a Basis for the Solution

4.4.1 Metadata

Metadata has been recognized as an important part of document organization systems and search systems, providing further information on documents for organization purposes or being utilized to enhance search results. Metadata about documents, in particular, has been used to help users understand more aspects of the documents, by search systems to improve search results and by classification systems to categorize documents.

The term “metadata” is used differently in different communities ([NISO Press, 2004](#)). The general agreement is that it is “data about data”, that is, data that describes other data. For purposes of this report it can be defined as “a description of the attributes of information objects that gives them meaning, context and organisation” ([Cornell University Library, 2003](#)). Metadata ensures that resources will survive and continue to be accessible into the future ([NISO Press, 2004](#)). Metadata has been classified into three broad categories: descriptive, structural and administrative. These categories do not always have well-defined boundaries, sometimes there are overlaps between them.

1. *Descriptive metadata* (also known as catalogue information ([Carr et al., 2001](#))) describes information resources for purposes such as discovery and identification. At the local (system) level it can be used to enable searching and retrieving, while at web level it enables users to discover resources. Elements such as title, abstract, author and keywords are part of descriptive metadata.
2. *Structural metadata* provides information about how compound objects are arranged to facilitate navigation and presentation of electronic resources. This can provide information about the internal structure of a document (e.g. sections, chapter numbering), the relationships among materials (e.g. photograph A in file B), and binding of related files and scripts (File A is PDF format of file B).

3. *Administrative metadata* provides information to help process and manage a resource. This includes technical data such as when and how a document was created and file type; rights managements, access control and use requirements; and preservation action information needed to archive and preserve a resource.

For desktop documents metadata comprises file attributes that are supported either as part of the filesystem, known as *built-in* file properties, or as an additional feature that allows users to define and associate files with metadata outside the filesystem (these are known as *extended file attributes*). For experiment purposes only the file properties available as part of the file system were used, but a discussion of use of extended attributes in the approach taken is discussed in section 8.2 as part of suggestions for further work. In Windows systems the file properties include *user-defined* values like filename (long file name specifies the path to the document based on the directory structure), author, keywords and comments, and *system-controlled* values such as creation date, last save time and number of pages. In addition some built-in properties can be defined and added by the user. These are known as *custom* properties. While system controlled values are mostly administrative metadata, structural and descriptive metadata are mostly found in the form of or encapsulated in user-specified attributes.

4.4.2 Tagging

The importance of tags for browsing content and enabling discovery of resources is now being extended to the desktop environment and is being applied to managing files and resources in this context. Originally tagging was applied in the web environment by online communities such as Flickr ⁴ and Delicio.us ⁵. Newer operating systems like Windows Vista and Mac OS X have incorporated tagging features as part of file and folder management. In Windows Vista, for example, a *tags* field has been added to the file *Save As* dialog for some types of files to enable users to specify tags for files.

Current work in the field of Personal Information Management includes the study of tagging resources, including documents, on the desktop in the context of the task the user is doing at that moment (Oleksik et al., 2009). With the realization that tagging is important for grouping resources and for flexibility of association, retrieval and switching between tasks the TAGtivity system (Oleksik et al., 2009) was developed to extend the utility of the file system. The authors were able to demonstrate that tagging can be effectively used to support users in accessing documents in the personal work environment. This was done with provisions for the definition of tags by users and for browsing of these tags, the resource metadata and thumbnails in Office 2007 applications, the Internet Explorer and the file system. WikiFolders (Volda and Greenberg, 2009) was

⁴www.flickr.com [last accessed 20/10/09]

⁵<http://delicious.com/> [last accessed 20/10/09]

also created to enable easy creation, editing and display of annotations with files and folders in the hierarchical file system after learning from the benefits of wikis in these aspects.

Tagging had also been previously applied in personal information management tools such as Phlat ([Cutrell et al., 2006](#)), Gnowsis ([Sauermann and Schwarz, 2004](#); [Sauermann et al., 2006](#)) and Presto ([Dourish et al., 1999a,b](#)). In these tools, as in newer operating systems, tags can be specified by users and are used to support retrieval through grouping of resources and filtering of search results rather than for promoting access and discovery of resources through browsing.

Users have already expended effort in classifying documents in folder hierarchies and views of folders as conceptual categories and document attributes have been expressed in literature. By creating the hierarchy users are already specifying tags or annotations for the documents which can be utilized as document index terms to provide a shorter and more precise path to the documents. These, together with other file attributes, have been seen to play an important role in helping users find and view documents in their hierarchies. File attributes which include the folder hierarchy (paths) already define their context and form a comprehensive framework to the hierarchy which can be exploited to provide exploration lines for reminding and helping users discover information in their personal document archives.

[Gonçalves and Jorge \(2004\)](#) list the attributes mostly used to describe documents as time, place, co-author, purpose, subject, other documents, format, exchanges, tasks, storage and contents. [Henderson \(2005\)](#) suggested that genre, task, topic or time be used as facets of document metadata as a result of the study she conducted. Table 4.1 shows the folder types used by the participants in the study. Genre, task, topic and time were the most used ones. Henderson also suggested that these facets be combined in different ways in navigation interfaces for personal digital document management. Most of the attributes described are already part of the document properties in the filesystem or are encapsulated in the folder hierarchy and the filenames.

[Henderson \(2005\)](#) compares the hierarchy of folders to defining a set of attributes and keywords for the document. Classifying documents into folders involved a major cognitive task ([Malone, 1983](#)), and because folder name attributes capture the multiple different roles that a single document might play ([Dourish et al., 1999a](#)), we can find ways of utilising them to provide for a more accurate associative structure.

Code	Description & Examples
Genre	Indicates that the contents of the folder are a particular class or type of document, with a commonly recognized form and structure. Examples: Lecture Notes, Presentations, Timesheets, Budgets, Letters.
Task	Indicates that the contents of the folder are related to a task, project, event or some other type of activity. Examples: Assignment 5, Lec01, PhD, recruitment, evaluation, For DSS Presentation.
Course	Indicates that the contents of the folder are related to a specific course. (This is a special case of Task above) Examples: Database Systems, 222, INFOSYS 222
Topic	Indicates that the contents of the folder are all about a particular subject matter. Examples: Web development, Database Architectures, JavaScript
Time	Indicates that the contents of the folder are related to a particular time period, or have a time related aspect. Examples: 2005, 2003 SC, Old, History, Week12, Archive
Person	Indicates that the contents of the folder are related to a particular person, group or organization. Examples: Matthew, Audit Committee
File Type	Indicates that the contents of the folder are all a particular file format. Examples: zips, PowerPoints, Excel docs
Temp	Indicates that the name of the folder appears to have no intrinsic meaning and that little thought was given to assigning the name. Examples: foo, bar, fffff, asdfasdf, New Folder
Source	Indicates where the contents of the folder originated, either a location or person. Examples: From Brenda, From J Drive, Copy of R Drive
Security	Indicates that the contents of the folder are subject to particular security constraints or permission level. Examples: Personal, Confidential, Private

TABLE 4.1: Folder types and their description (Henderson (2005))

4.5 Summary

The problems of locating and relating documents on the desktop has been presented. The need for a method of dynamically organizing documents that aids retrieval, and offers reminders and associative browsing has been clarified. Evidence points to the importance of user-specified attributes in helping to realise this goal. A review of information-seeking interface design provides insight on what methods to adopt for the solution and what aspects to consider for provision in the interface. Browsing as opposed to search has been identified as an effective solution in exploration tasks. Browsing offers flexibility in retrieval without the need for specification of needs and utilization of existing information, especially user-specified information. It can therefore help in easing user memory load by exposing information that can act as effective reminder and retrieval aids.

The next chapter looks specifically into sourcing, structuring and use of document attributes in formulating a solution for effective retrieval, association and discovery of documents on the desktop.

Chapter 5

Metadata Specification and Experiments

5.1 Introduction

This chapter describes the use of document metadata in formulating a solution for the problems of retrieval, association and aiding of discovery of desktop documents. Document properties form the basis of these solutions and are extracted from the filesystem, structured and stored as semantic data. Structuring the metadata in Semantic Web form enables easy storage, future extensibility with other information sources on the desktop and access from different applications. An explanation of how the encoding was carried out including the definition of an ontology and specification of URIs for files on the desktop is included. The chapter also describes the experiments conducted to determine how this semantic metadata can be used to solve the problems specified. The semantic document metadata is used in the experiments to explore links within it and with other related semantic data sets. The methods employed are then critically assessed with regard to the challenges encountered and how far they solve the inadequacies specified.

5.2 Metadata From Desktop Documents

5.2.1 Microsoft Systems' File Properties

As previously mentioned documents on the desktop contain rich metadata in the form of documents' properties stored together with the documents. Some of these properties are shared across applications while others are application-specific. The generic properties include the file name, type of file, location, size, dates the file was accessed, created and modified, and others that can be modified by the user namely title, subject, author,

category, keywords and comments. On Windows systems these can be viewed and editable ones edited either through the right-click menu on the desktop (see figure 5.1) or through the specific documents' applications. Application-specific properties are those that are relevant only to particular applications, for example Worksheets in Microsoft Excel.

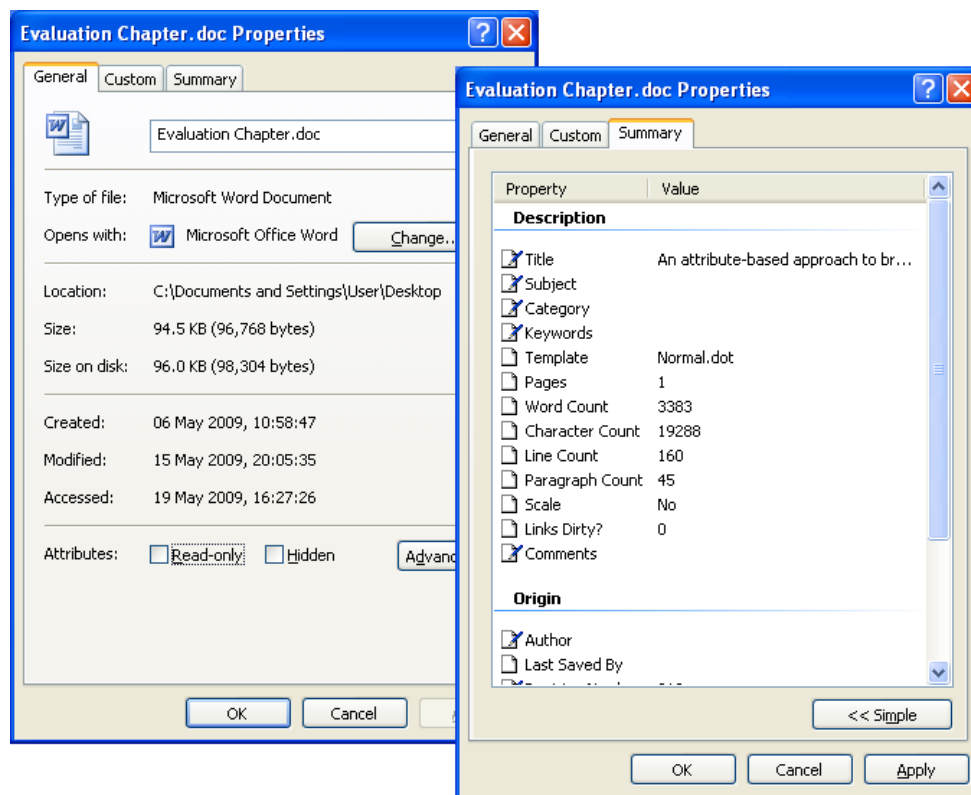


FIGURE 5.1: Generic built-in file properties for a Microsoft Word file in Windows XP

5.2.2 Extracting File Properties in Windows

File Properties in the Microsoft Windows environment are stored as part of the filesystem and all are available to read from. In addition all but the system-generated/controlled ones can be added or edited. These built-in document properties can be accessed programmatically through the *Windows Management Instrumentation* (WMI) commands. The commands require technologies such as the Component Object Model (COM) to enable a program to interact with the Windows Operating System through languages like Microsoft Visual Basic, C++ and scripting languages on Windows that can handle Microsoft ActiveX objects such as VBScript. The architecture of WMI and its interaction with languages and resources is depicted in figure 5.2.

At the top of the infrastructure are the WMI consumers; scripts, applications or tools that access and control information available through the WMI architecture. The three WMI primary components, namely the Common Information Model Object Manager

(CIMOM), the Common Information Model (CIM) repository, and the providers, provide the infrastructure through which the configuration management data is defined, exposed, accessed, and retrieved. The CIM object manager describes the exposed management information while the CIM repository stores the management data. The WMI provider acts as an intermediary between the CIMOM and the managed resources. The managed resources are the logical or physical components that are exposed and manageable by WMI ([Microsoft Corporation, 2009b](#); [Boshier, 2000](#)), in our case files. Through the resources' application programming interfaces (APIs) WMI can access and control these resources.

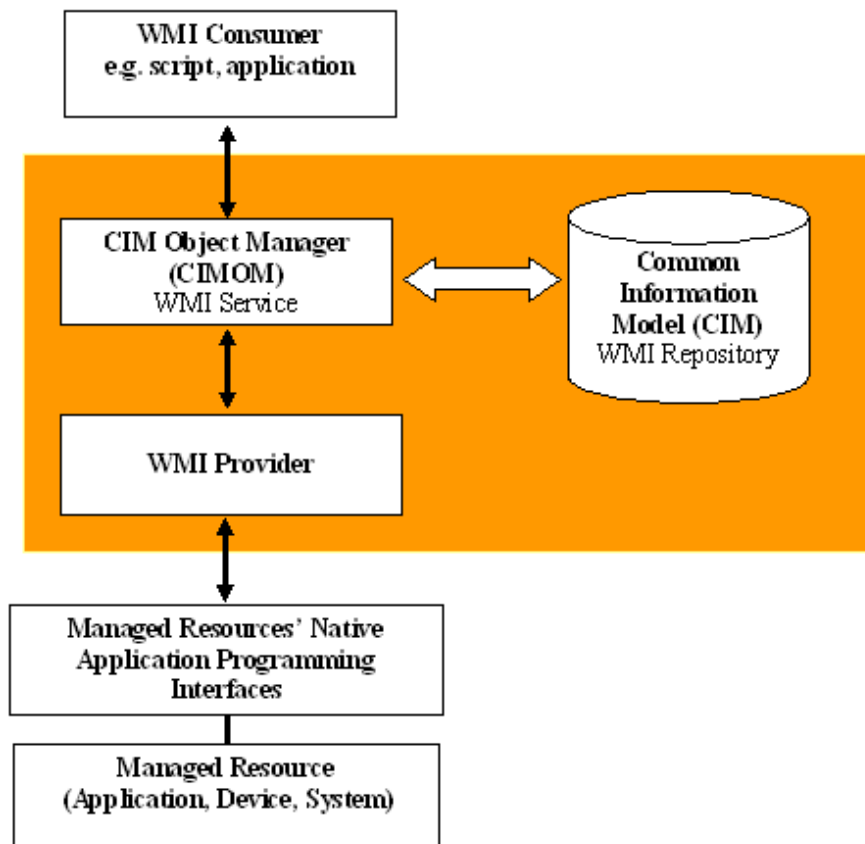


FIGURE 5.2: WMI architecture and its interaction with other components

For this thesis, for matters of practicality and sampling, we extract all properties for commonly used documents types and only the filename and location for all other files. The commonly used document types we concentrate on are Microsoft Office documents (Word, Excel and Powerpoint), html, htm and portable document format. Visual Basic Scripting Edition (VBScript) is used as a lightweight approach to implement the WMI classes, with the advantage of its ability to utilize the Windows Scripting Host to run directly on a user's computer.

WMI commands are used to crawl the hard drive within a given root or a given top-level directory to recursively explore the subfolders and files contained therein in turn. For each file the built-in properties are extracted in the form of property-value pairs.

5.3 Encoding and Storing the Metadata: RDF and the Semantic Web

To make it easy to identify the files, properties and values extracted and relate them the metadata is structured into semantic web form (using the Resource Description Framework - RDF) based on a defined ontology.

5.3.1 The Semantic Web

The Semantic Web, an initiative conceived by Tim Berners-Lee ([Berners-Lee, 1998](#); [Berners-Lee et al., 2001](#)), proposed an extension to the World Wide Web (WWW) such that data on it could become logically connected. To achieve this he proposed that data on the web be restructured from human-understandable form to a machine-processable form so that computers could also participate in the sharing of information. With the Semantic Web collections of information can be structured and given well-defined meaning, enabling both people and computers to better manipulate and share the information.

Two technologies, the eXtensible Markup Language (XML) and the Resource Description Framework (RDF) are used to structure the collections of information (documents) and express the meaning of the data (metadata) respectively, encoding it to form webs of information about related things. Individual things are identified by Uniform Resource Identifiers (URIs), and relationships between the things encoded in RDF in triples of the form subject, predicate, and object. The semantics of RDF documents are represented in a structure called an *Ontology*. The ontology specifies at least a vocabulary of terms and some specification of the meaning of those terms ([Carr et al., 2001](#)).

5.3.2 The Resource Description Framework

RDF is an infrastructure used to express information about resources. Although the word “resource” has been used mostly to mean web resources, Hypertext Markup Language (HTML) documents and multimedia files on the web, it can also be used for things from the real world, such as products or persons. These resources have properties and are uniquely identifiable by an URI ([Miller, 1998](#)). RDF expresses facts about these resources using statements, each statement containing a subject resource, a relationship

(property-type or predicate), and an object (property-value) which itself may be a resource. RDF data can be stored in files or databases and is designed to be published on the web. A description is a collection of these properties that refer to the same resource. A generic RDF Description is shown in figure 5.3 (Gruber, 1995).

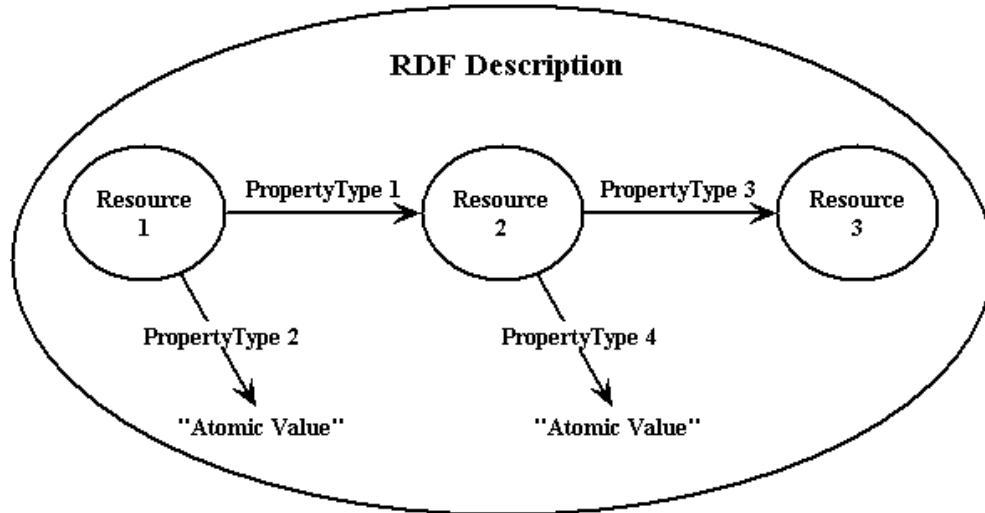


FIGURE 5.3: Generic RDF description

An example of the statement “the author of the file *references.doc* is Jane” depicted graphically as a data model is shown in figure 5.4.

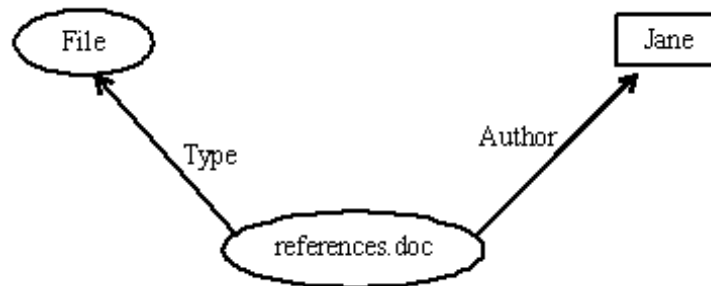


FIGURE 5.4: Data model of example statement

5.3.3 Defining an Ontology

An Ontology can be defined as “an explicit specification of a conceptualization” (Gruber, 1995). A conceptualization defines the objects, concepts and other entities of interest in a domain of interest and the relationships that hold between them. The types of the concepts used and the constraints on their use are explicitly defined. An ontology therefore defines this view of the world that is represented for a particular purpose. By defining the meaning of metadata terms, ontologies allow the semantics of documents

to be used by web applications and intelligent agents, facilitating easier integration and sharing of information.

For this research an ontology is designed to define the domain of desktop files in order to be able to represent metadata about files on the desktop in a concise and identifiable way. Some terms are reused from standardized ontologies like the Dublin Core and languages such as the RDF Schema (RDFS) for defining the metadata and the ontology specifies the additional ones that could not be found in the commonly existing schemas. The ontology was defined using SWOOP version 2.2.1 ([MINDSWAP Research Group, University of Maryland, 2004](#)), a tool for creating and editing Web Ontology Language (OWL) ontologies.

5.3.4 Defining URIs for the Files

URIs for files on the desktop in this setup are made by appending the prefix “file:///” to the full name with backslashes replaced by forward slashes. The computer identity is not included in the URI since the data and implementation are for personal use and will not be requested over the network. This was done following the specification RFC 1738 ([Berners-Lee et al., 1994](#)) which describes Uniform Resource Locators (URLs), strings that allow for location and access of resources via the Internet. A URL is in the form

`file://<host> <path>`

where *host* is the fully qualified domain name of the system on which the *path* is accessible, and *path* is a hierarchical directory path.

The character “/” is used where the host name is supposed to appear, and no internet protocol or access method is specified. Also characters that are used in URI parsing and those that are allowed in Windows file names but not in URIs are percent-encoded. The URI provides a direct link from the metadata, and therefore from the implementation, to the document.

5.3.5 The Semantic Desktop File Metadata

An RDF description of each file based on the built-in properties is then defined based on this ontology and stored together with that of other files in a RDF file. This semantic data about desktop files’ metadata enables easy integration of the metadata itself. The data can also be accessed, manipulated, interacted with and explored from a variety of applications. The Ontology structure is shown in figure 5.5 and its definition is included as part of this report in Appendix B. An example of an RDF description for the file “pgr-rtsg.doc” on the researcher’s desktop is shown in figure 5.6. “MSFilemetadata” denotes the created ontology’s base name.

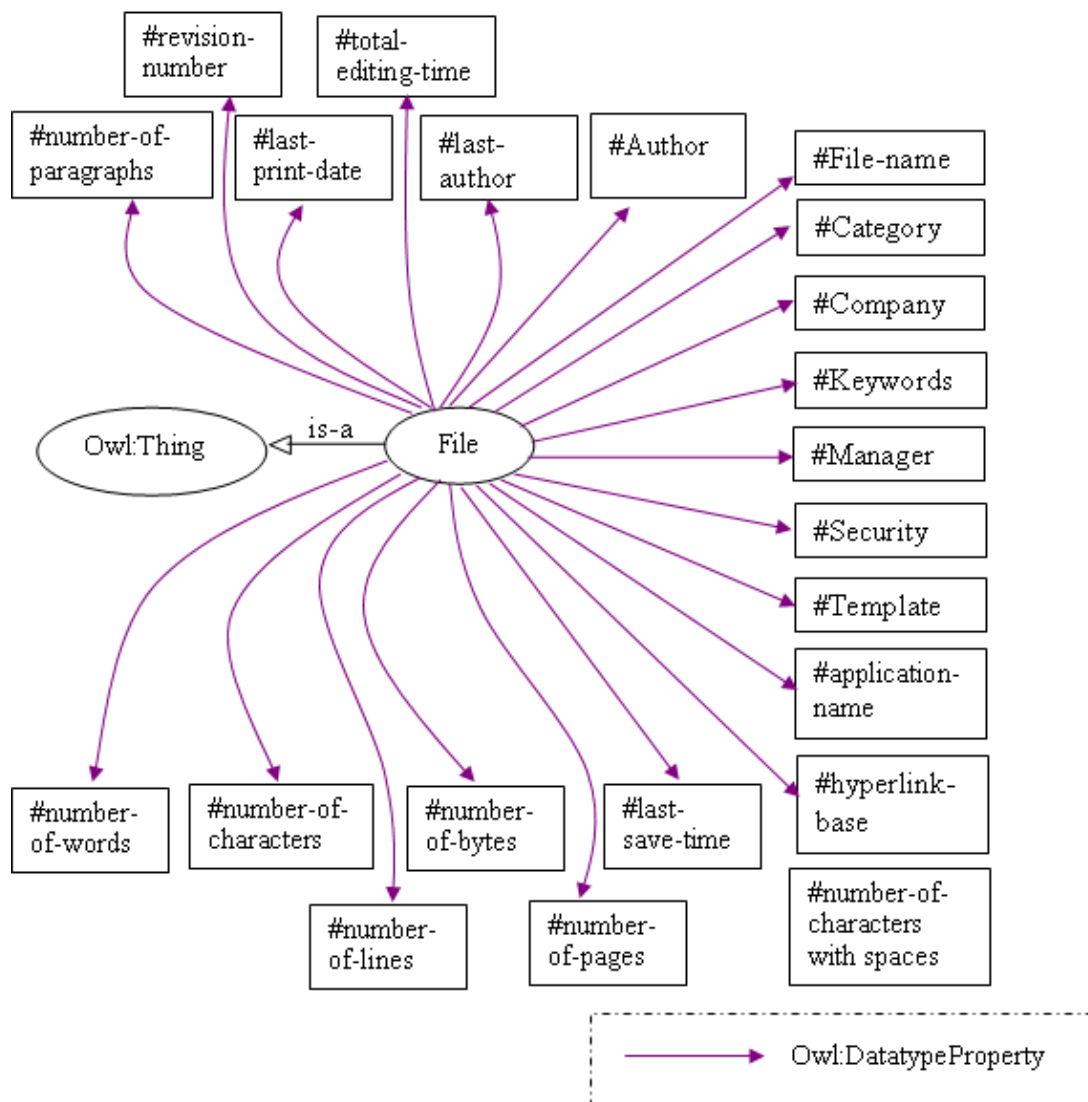


FIGURE 5.5: Overview of the file ontology

Through VBScript, we utilize the FileSystemObject (FSO) object model and automation objects in Windows to programmatically create, write to and save files and folders as the built-in properties are collected from the file hierarchy. Incomplete metadata like the one shown are common in desktop files but are still very useful as discussed in the next section.

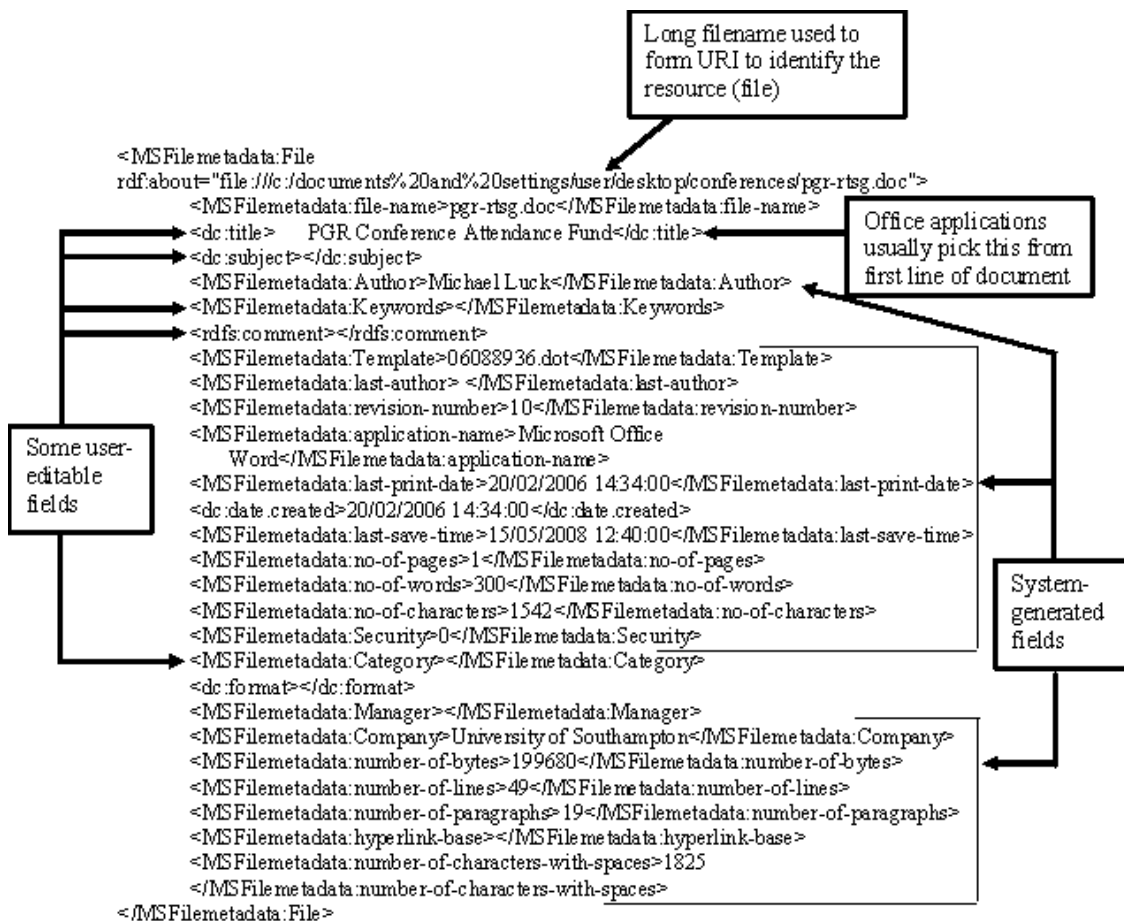


FIGURE 5.6: An example of a RDF description for a file

5.4 Experimenting with Specification and Filtering Using Attributes

5.4.1 Specification of File Metadata in Windows

In Microsoft Office 2003 productivity applications (Word, Excel, Powerpoint etc), metadata can be added to a document in at least two ways. The first way is when the file is open by using the application menu and selecting *File* then *Properties* to view and edit the file properties. The second way is through the file icon on the desktop by right-clicking on the icon and selecting *Properties* from the pop-up menu. Though these interfaces are built to serve a good purpose there is not enough effort made to prompt and encourage the user to add metadata, they have to remember to add it either while they are still editing the document or after saving it. Office 2007 came with an effort to counteract this. Users can configure the applications such that they can be prompted for these file properties through a pane when they save a file. The method used to get to the properties while still editing the document has also been changed. The properties are now available through the *Office* Button then selecting *Prepare* then *Properties* on the

sub-menus. But the benefits of adding this metadata are still not immediately obvious to the user, especially for grouping and associating files.

The process of metadata specification is similar to tagging, which has become widespread and very important in applications especially on the web. Both serve the same purpose of facilitating organization and discovery of resources. Metadata in the form of attributes have values attached to specified attributes while tags are free-form keywords attached to the resource itself. Researchers like [Cutrell et al. \(2006\)](#) have recognised the importance of tagging and predicted its widespread in the future which might even replace filing systems. They have also foreseen that if tagging and associated systems are not supported throughout users workflow, the usefulness of such systems might not be realised.

Based on these observations a small C# application was developed to encourage users to add common attributes to the files as they save them. In the Microsoft Windows *Save file as* Dialog, two more controls were added below the *Save as type* combo box. These were labelled *Property* and *Value*. The Property combo box allows the user to choose to add one of *Subject*, *Category*, *Comments* or *Keywords* attributes. The Value text box allows the user to type the text for the attribute chosen in the combo box. Figure 5.7 shows the keywords “Useful Stuff about C# programming” being added to the file *programming tips.doc*. The metadata is saved with the file as part of the filesystem as the user clicks on the *Save* button. By adding such similar metadata to a set of files users are explicitly specifying connections between the files.

5.4.2 Filtering Documents Using Metadata

The metadata was then used to filter the files being which contain a specified attribute with the given attribute value. By accessing the filesystem-stored property names and values on demand inside the application the program allows the user to filter the file-names based on the metadata when opening the files using the Microsoft Windows *File* → *Open* Dialog. The files matching the criteria are presented on selection of a property and specification or selection of a property value on a selected folder. Figure 5.8 shows how to view files with the “category” attribute value “research”. If the user has a lot of files in the same folder, this can greatly reduce the number of files offered to choose from, thereby helping the user to easily locate the file they are looking for.

This approach, while being simple and demonstrating the value of metadata for associating documents and helping users by reduce effort needed to retrieve similar documents, is obviously inadequate and presents some challenges both in specification of metadata and the association presentation interface.

1. The metadata values are uncontrolled, that is, there are no restrictions and enforced ‘sense’ on the values and they do not necessarily come from a controlled

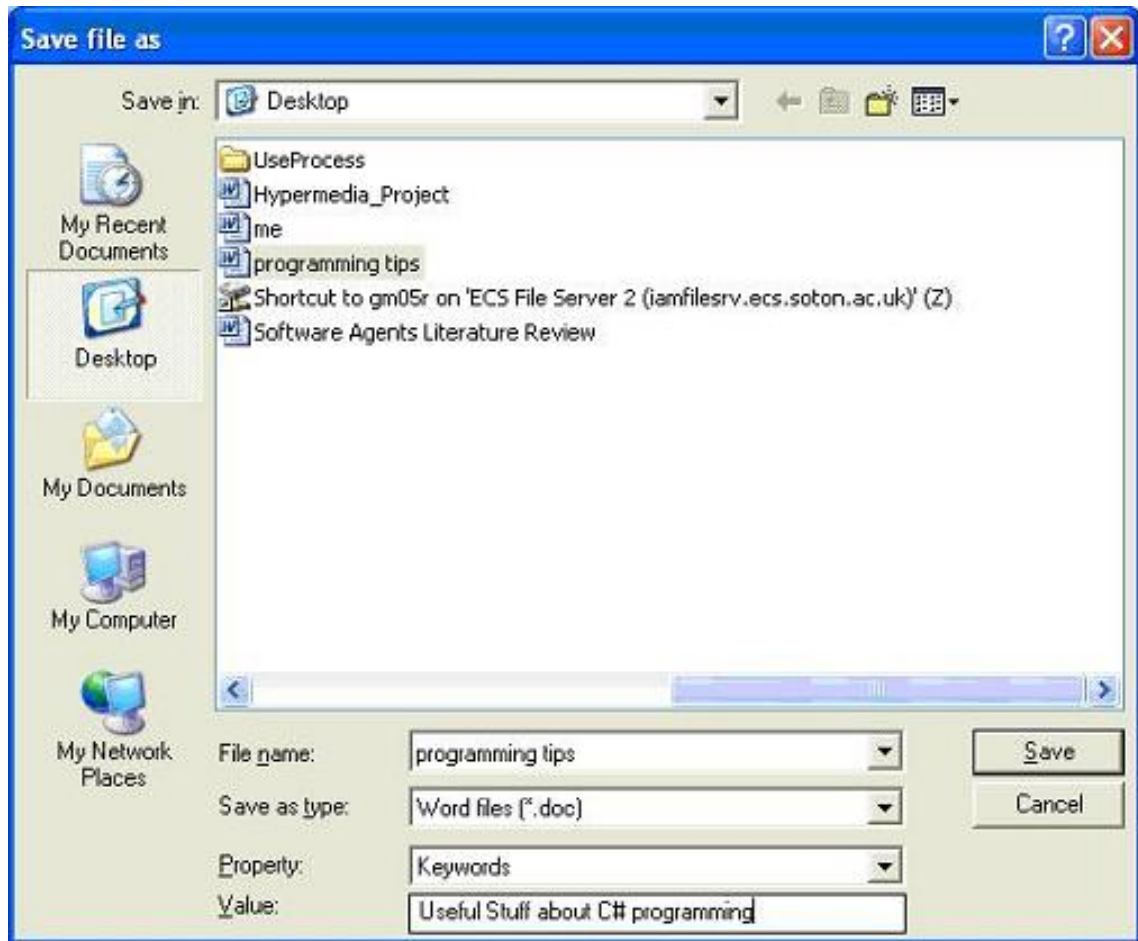


FIGURE 5.7: Adding value for attribute while saving a Microsoft Word document

vocabulary as recommended by metadata standards. As such the benefit of differentiation of this metadata for fields such as keywords, subject and category might not be realized. The user is given categories in which to specify the metadata values, which can be single or multi-worded (see figure 5.7 and figure 5.8). While these categories can be important to both the system and user it might be difficult for the user to use consistent categories and wording throughout the different documents. Assistance can be provided through presentation of already-existing values as the user specifies the values, but filtering is still based on a particular property and its value.

2. The use of specific categories again presents problems in associating documents across different applications and folders. Research shows that users have a problem with this and applications implementing methods enabling grouping of files and specification of metadata once for the whole group have already been implemented to counteract this, an example being an application developed by Getz (2006). The application allows setting of Microsoft Word properties for a whole folder to be assumed for all documents in that folder.

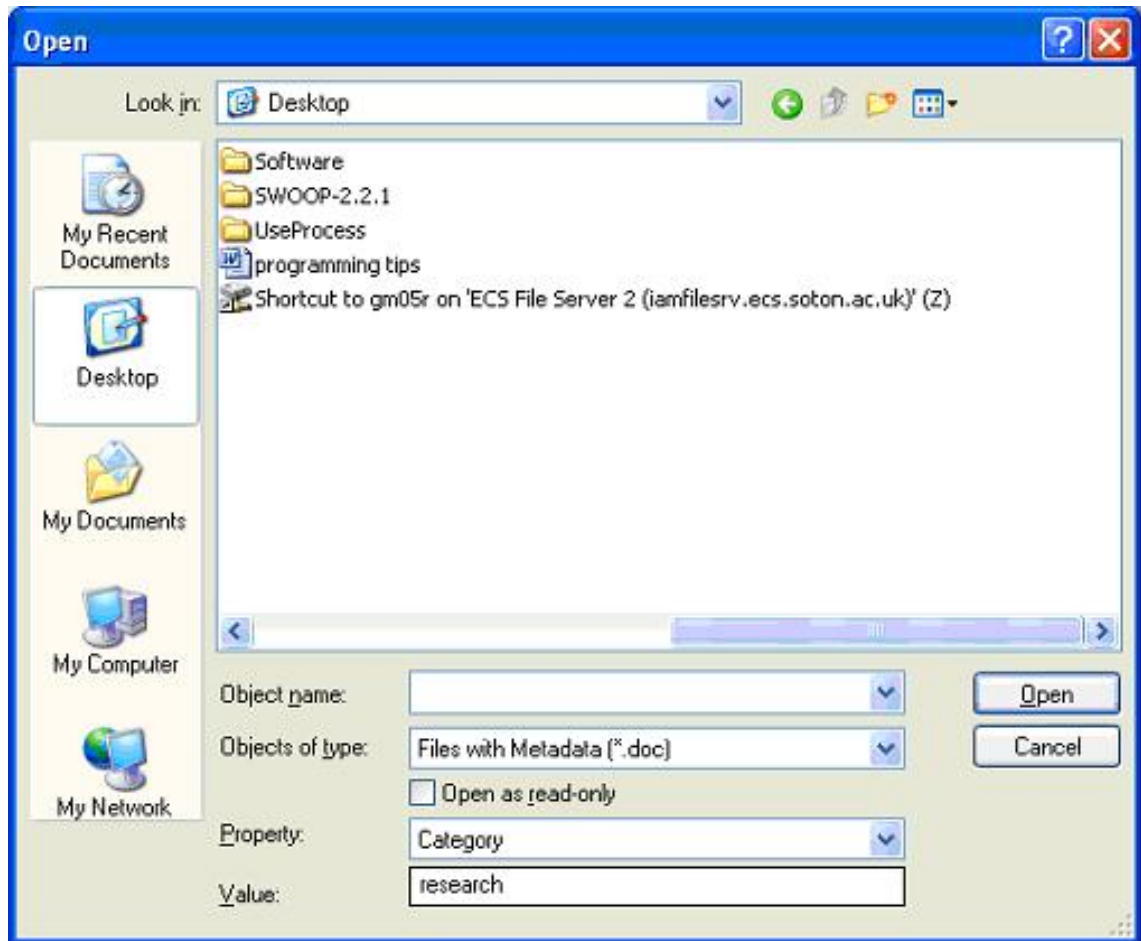


FIGURE 5.8: Selecting Microsoft Word documents with specified metadata

3. To get the benefit of this method users are required to open the files through the *File* → *Open* dialog inside a specific application. Because there is another simpler method of opening files (double-clicking on a file icon) this method might fail to sell and therefore be utilized if it is not the users' regular way of opening files.
4. Comparisons to determine clustering are so far based on exact matches of values. Enabling the addition of multi-worded metadata values could however benefit the user more if some data processing is done to show partial matches based on content of the values in addition to full comparison.

Lessons learned using the described approach are taken further to investigate use of semantic metadata as a solution to connect and present documents on the desktop and integration with other associated information from semantic datasets.

5.4.3 Browsing Semantic Metadata

Data structured in RDF form can be interlinked by matching equivalent URIs. Semantic Web browsers provide an interface to explore this semantic data. These enable users to

browse from resource to resource. The Tabulator⁶ is a generic linked data browser that was developed to help users traverse their RDF data and discover and use data that they had not thought of using (Berners-Lee et al., 2006). Navigation is provided along relationships (predicates) in a web of concepts. Multiple visualisations are provided with exploration provided in outliner mode, and map, calendar and timeline views are also available (see figure 5.11). Users can also construct RDF query patterns which are then translated into the SPARQL language specification.

The outliner mode, which presents relationship between resources as a tree was used in an experiment to investigate how it could be used for browsing the documents' metadata. The tree-view was adopted because it is a common user interface metaphor and also provides a denser tabular format to enable more information to be viewed (Berners-Lee et al., 2006), while preserving the browsing path during navigation and thereby keeping an overview of the collection. Links in Tabulator are expressed in both directions, providing more exposure and paths to data. Tree-oriented browsing is also used in other Personal Information Management(PIM) tools such as SEMEX (Cai et al., 2005) and Gnowsis PIMO (Sauermann and Schwarz, 2004).

Figures 5.9 and 5.10 show browsing the semantic file metadata in Tabulator using the outliner view before re-engineering. Titles, where available, are shown in place of file-name and opening a "file" resource shows the properties and values which can be used for query-by-example to select and view similar items.

The rest of this section presents the implementation based on the extending (re-engineering) the outliner view of Tabulator to explore semantic document metadata. Other linked data is then incorporated into the interface to check whether the document metadata can be useful for associating and exploring appropriate resources in a given domain.

5.4.4 Adopting and Reengineering Tabulator Modules

Code from the Tabulator is available as open-source AJAR (Asynchronous Javascript and RDF) scripts, which utilizes interfaces including Hypertext Markup Language (HTML) Document Object Model (DOM), Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML), RDF and SPARQL. The scripts run on a web page to add data browsing and were downloaded and reused to create the browser for the experiment. The 0.8 version release was used for the experiment.

The open-source code consists of two sets of modules (Decentralized Information Group, MIT, 2006). One set, the AJAR library, provides data handling methods like web access, store and query while the other set (figure 5.11), the Tabulator modules, provides the various views supported. The AJAR library and the outline mode were adopted in this

⁶<http://dig.csail.mit.edu/2005/ajar/ajaw/tab> [last accessed 11/08/09]

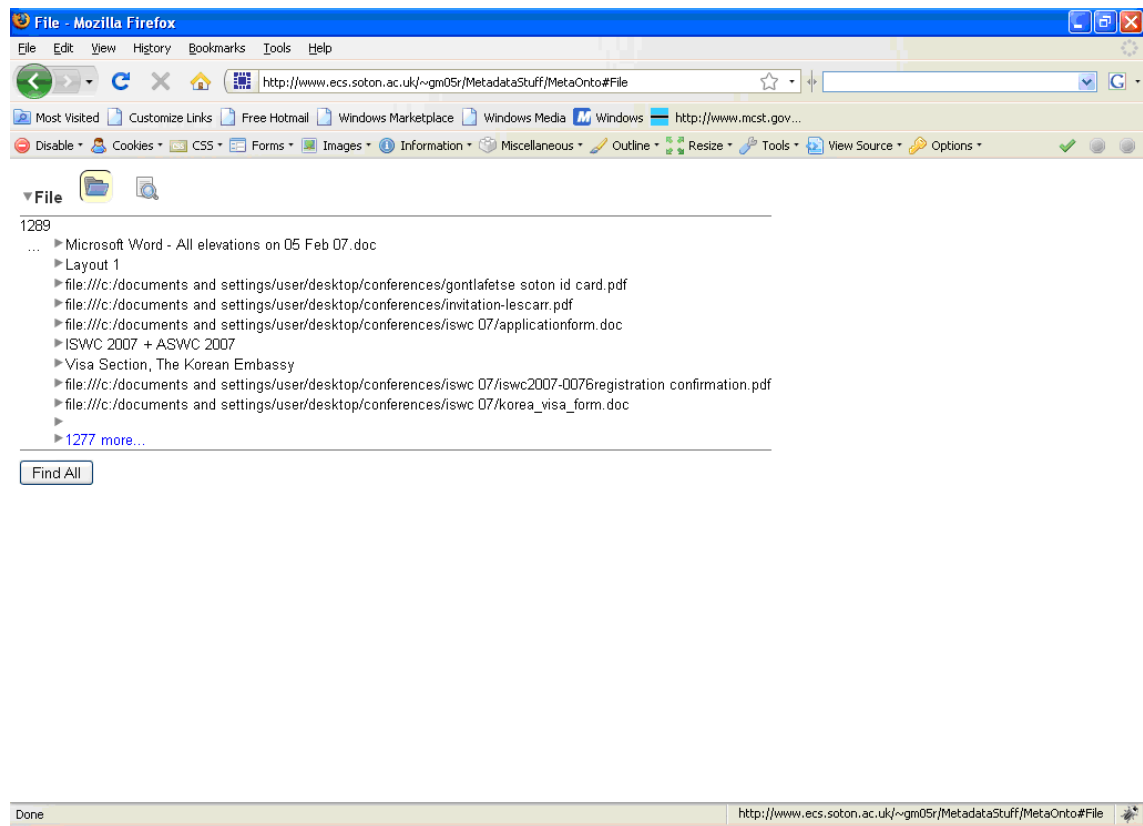


FIGURE 5.9: Browsing the metadata using the Tabulator Extension 0.8.7

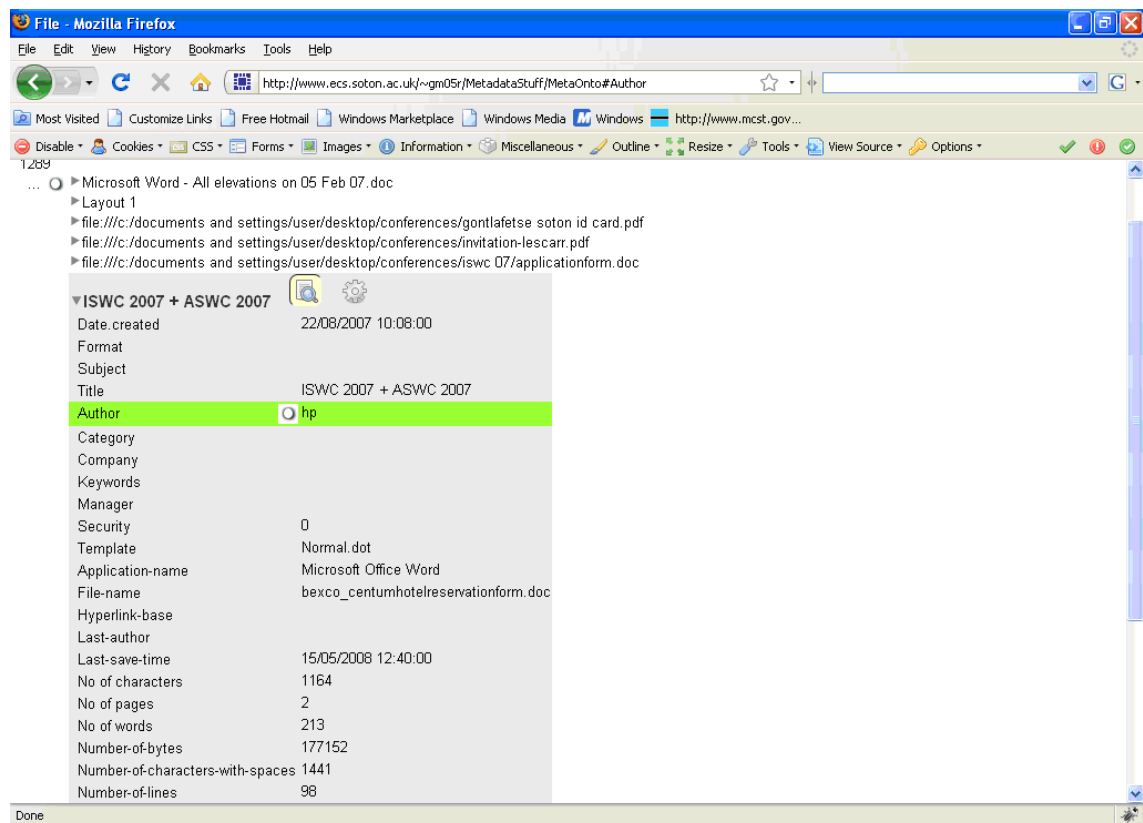


FIGURE 5.10: Browsing properties and values

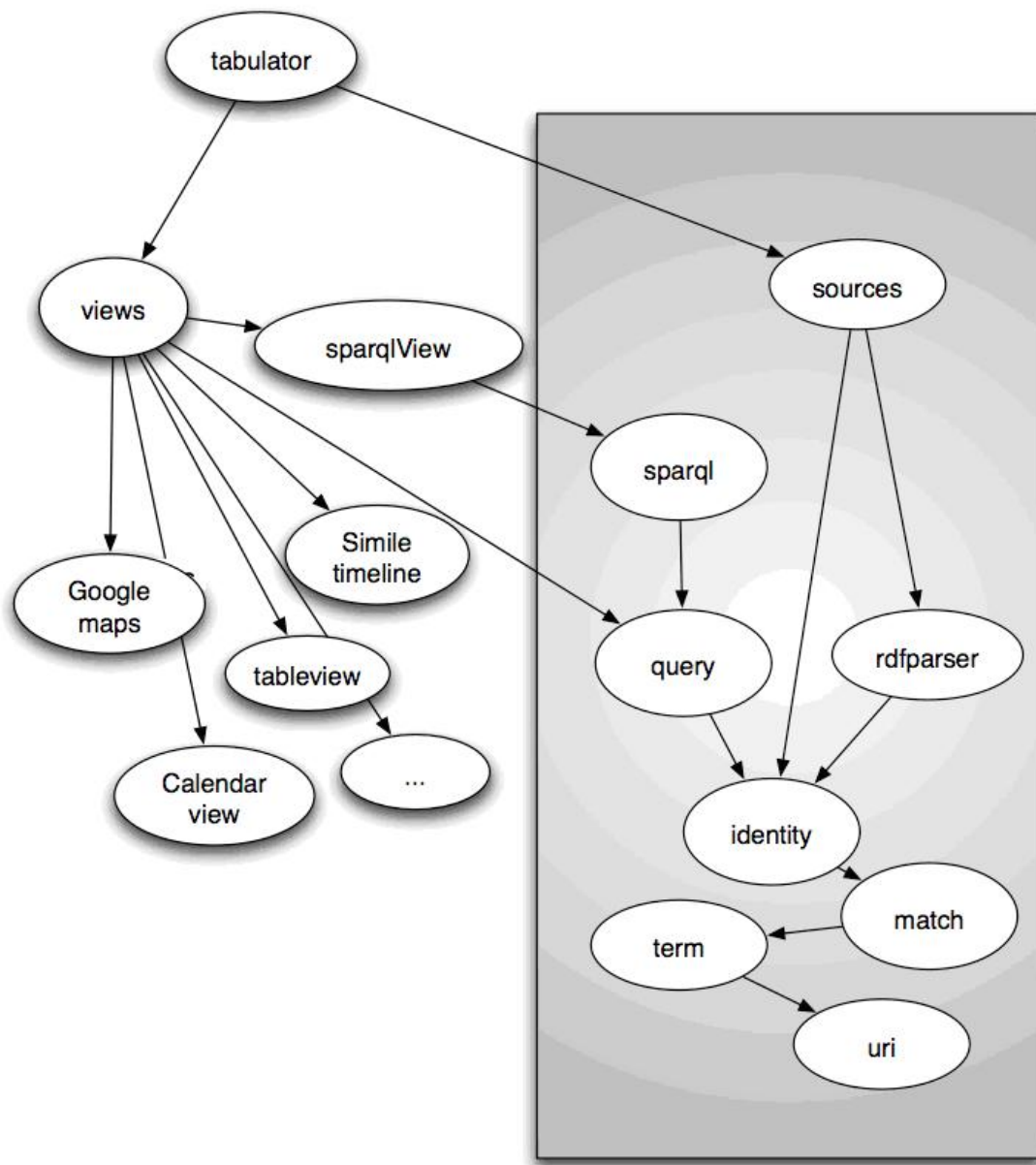


FIGURE 5.11: Dependency of modules in Tabulator and the AJAR library ([Decentralized Information Group, MIT \(2006\)](#))

project. An RDFParser from the AJAR library is used to parse RDF/XML files and store them as triples in the form of subject, predicate and object in its knowledge base. The RDF triples can then be easily accessed or queried using the SPARQL modules provided. No installation of programs and database backends is required, hence making the code easier to adopt for experiment purposes.

Data in the form of the file metadata created was loaded into the program.

5.4.5 Browsing, Query-by-example (QBE) and File-Open Interface

The interface was re-designed to present the metadata for each document as individual resources with the file name as a description (see figure 5.12). In a similar method to that used in Tabulator, files can be opened from the interface (using the icon in front of the filename), just like Uniform Resource Identifiers (URIs) which can be looked up for web resources, except that the files are opened in their own application or the one the user chooses rather than in a web browser.

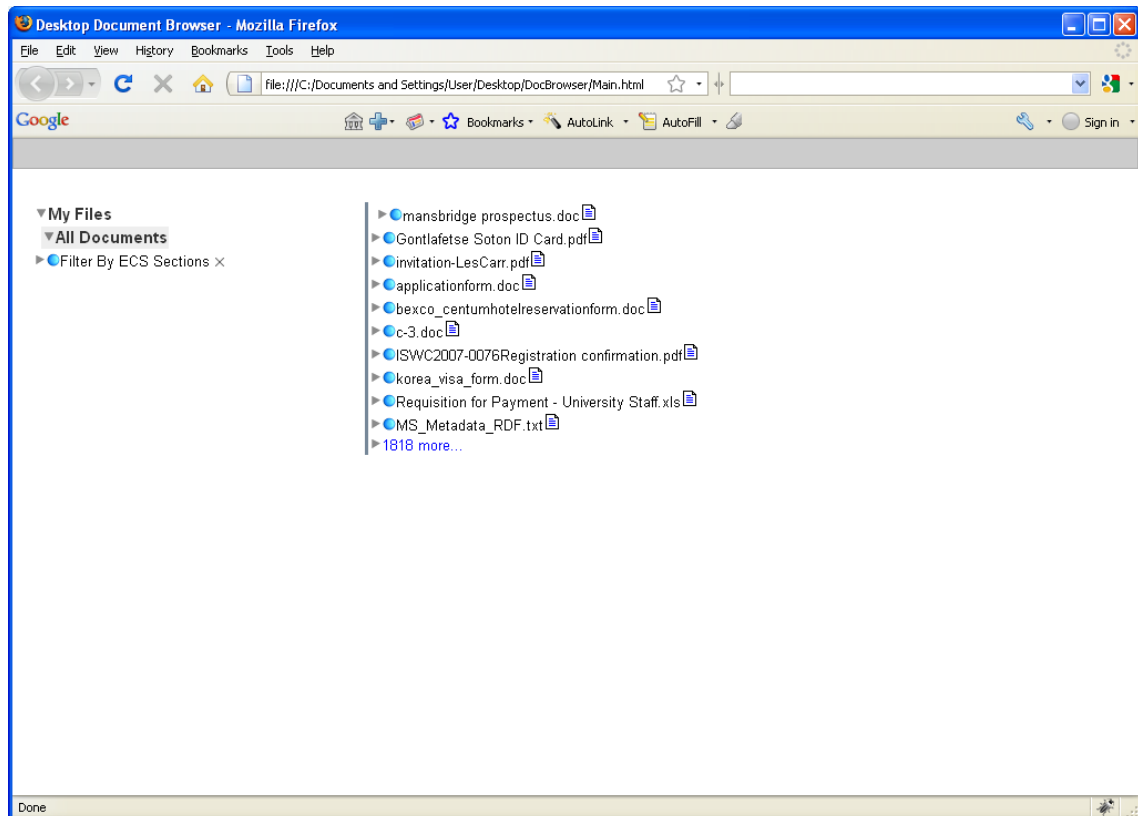


FIGURE 5.12: Documents metadata as resources

Expanding the file name node reveals the metadata for that file. The metadata can be queried for similar items based on the attribute value by selecting the icon in front of its value (figure 5.13). By so doing the user can browse implicit associations between the documents based on these attribute values in a QBE form. Documents can thus be re-organised according to the document properties for the task at hand ([Decentralized Information Group, MIT, 2006](#)). Tabulator itself provides QBE by enabling the user to highlight a property field to construct a query pattern based on the series of edges connecting the field to the root node ([Berners-Lee et al., 2006](#)). The results of such a query are provided in a separate table from the outliner.

Another aspect that was looked into was browsing with an option of filtering by date (figure 5.14). Also with availability of URIs in the property values also greatly impacts the browsing experience with more useful connections visualized (figure 5.15). URIs for

▼ **Requisition for Payment - University Staff.xls**

date.created: 05/05/2004 10:22:14 [similar items ▶](#)

format: [similar items ▶](#)

subject: [edit](#) [similar items ▶](#)

Author: Department Of Electronics [edit](#) [similar items ▶](#)

Category: [edit](#) [similar items ▶](#)

Company: University of Southampton [similar items ▶](#)

Keywords: [similar items ▶](#)

Manager: [similar items ▶](#)

Security: 0 [similar items ▶](#)

Template: [similar items ▶](#)

application-name: Microsoft Excel [similar items ▶](#)

file-name: Requisition for Payment - University Staff.xls [similar items ▶](#)

hyperlink-base: [similar items ▶](#)

last-author: [similar items ▶](#)

last-print-date: 20/11/2007 18:13:29 [similar items ▶](#)

last-save-time: 20/11/2007 18:23:02 [similar items ▶](#)

revision-number: [similar items ▶](#)






















comment: [edit](#) [similar items ▶](#)

Manager: [similar items ▶](#)

Security: 0 [similar items ▶](#)

Template: [similar items ▶](#)

application-name: Microsoft Excel ▼

- ▶ Requisition for Payment - University Staff.xls 
- ▶ Feedback_Results.xls 
- ▶ HotelQuotes.xls 
- ▶ Feedback_Results.xls 
- ▶ Feedback_Results_Oct07.xls 
- ▶ UB ITD - 2008 - CURRENT - MAY 2008.xls 
- ▶ SALARY SCALES April 2008 (2).xls 
- ▶ tax.xls 
- ▶ 4 Open Repositories 2008 - rooming list.xls 
- ▶ Accommodation Issues.xls 
- ▶ Accommodation OR08.xls 
- ▶ Copy of 4 Open Repositories 2008 - rooming list 18-03-08 (3).xls 
- ▶ Debits-Credits 18-04-08.xls 
- ▶ Debits_13-03.xls 
- ▶ Debits_26-02.xls 
- ▶ Debits_26-03.xls 
- ▶ Debits_27-03.xls 
- ▶ Jury's Inn Booking.xls 
- ▶ local.xls 
- ▶ UB ITD - 2008 - CURRENT - MAY 2008.xls 
- ▶ Results Depth.xls 

file-name: Requisition for Payment - University Staff.xls [similar items ▶](#)

FIGURE 5.13: Query-by-example embedded in interface in the re-engineered browser

entities that have been organized in RDF can be looked up in a table if such a table has been constructed with knowledge from the domain. For personal data, this is usually varied and makes it difficult to do so except for fields that deal with domain-specific knowledge, mostly organizational data.

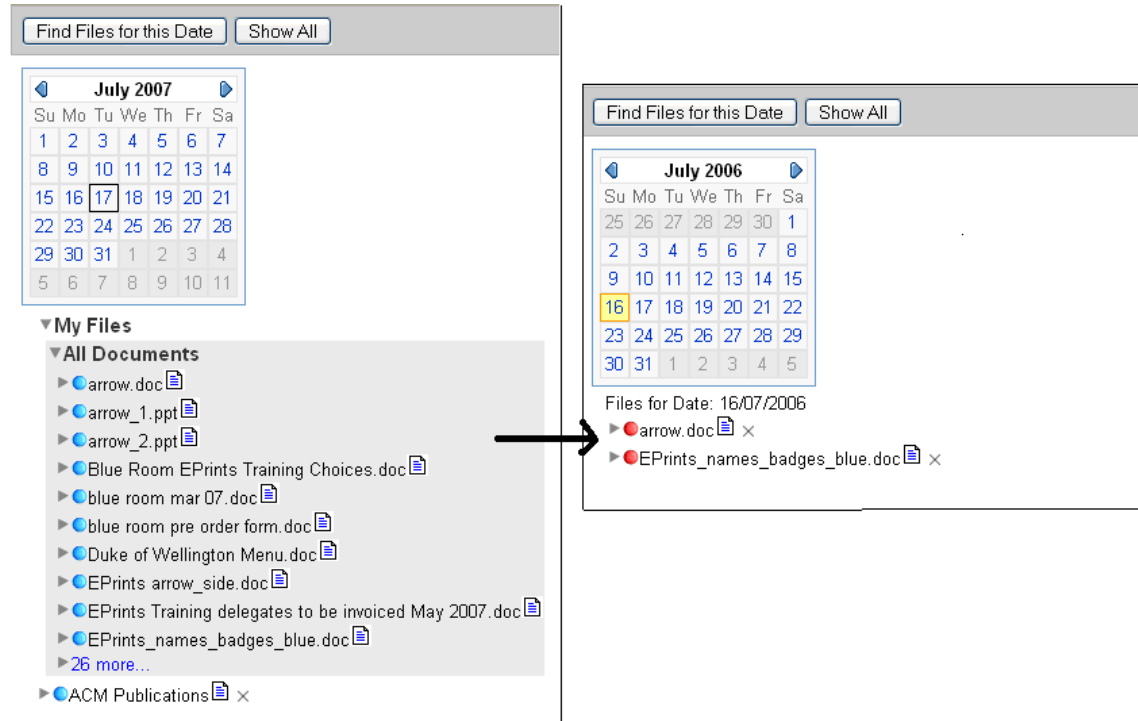


FIGURE 5.14: Selecting a date filters document list to include only relevant results

Enabling viewing of QBE results inside the outline provides more smooth browsing and exploration for re-organizing or grouping similar documents based on factors such as author, time of creation, and application which may not be in the same folder in the hierarchical file system. By providing such an associative and dynamic re-organizing facility the user can be assisted in viewing connections between their data with minimum effort.

This method is similar to that used in the SEMantic EXplorer(SEMEX)⁷ as seen in figure 5.16.

5.4.6 Adding and Editing Attributes in the Interface

Although the system is primarily for browsing documents, mechanisms are also provided to add or edit attributes such as category, subject, keywords, title and comments (figure 5.17). These are the built-in document properties that can be edited by the user and serve as simple annotations to the documents. Though users are known to be reluctant to add metadata to their documents (Kao et al., 2003), an opportunity to add them

⁷<http://db.cs.washington.edu/semex/semex.html>, last accessed 14/08/09



FIGURE 5.15: Browsing property values which are URIs gives more information

should be availed for them to do so whenever possible given the importance of user annotations.

5.4.7 Connecting to Other Semantic Information spaces

The advantage of acquiring and using the Tabulator scripts on one's desktop lies in that semantic data sets can be loaded and browsed on demand. To experiment with the browsing of desktop data in connection with RDF datasets the parser was further loaded with RDF data from the School of Electronics and Computer Science(ECS), University of Southampton ([Southampton ECS, 2007](#)). ECS RDF data is available in file sets as groups (research, commercial and other) which consists of entities such as themes, projects, people and seminars. Other entities like interests, locations and degrees can be sourced by expanding these components, or by extracting them as single entities. These

Browsing Associations with Semex

SEMEX User Info

Keyword: **LUNA DONG**

AuthorOfArticles

MentionedIn

SenderOfEmails

RecipientOfEmails

Coauthors

Reference reconciliation identifies all references to the same real-world object

Different references to the same person

FIGURE 5.16: Browsing with SEMEX

```

▼ arrow.doc
creator: ▶ http://id.ecs.soton.ac.uk/person/7037
date.created: 16/07/2006 23:58:00 ▶
format: ▶
subject: Directions  save
Category:  edit
Company: University of Southampton ▶
Keywords: ▶
Manager: ▶
Security: 0 ▶
Template: Normal.dot ▶
application-name: Microsoft Office Word ▶
file-name: arrow.doc ▶
hyperlink-base: ▶
last-author: ▶ http://id.ecs.soton.ac.uk/person/9708

```

FIGURE 5.17: Editing the “subject” attribute

data entities (people, projects, interests, themes and so on) can correspond to document metadata for a person belonging to the same organisation and as such associations can be deduced between the two. The Document Browser implemented allows the user to filter or sort their documents according to these entities.

To derive explicit associations automatically a few commonly-used methodologies are used. These categorisation methods are not exhaustive and may also come up with wrong results (see discussion later). The following process is carried out when the user selects an entity to use to filter documents from the ECS entities listed.

1. The knowledge base is queried for all objects of the selected entity type, and these are kept in memory for subsequent comparison with document metadata. The set of documents with the corresponding full filenames (name including directory information), properties and their values are extracted for comparison with the above such that associative integration can be performed.
2. The document attribute values are divided into words which are then searched (queried) against the last part of the entity (name or value) chosen as a filter, that is, minus the prefix - (<http://rdf.ecs.soton.ac.uk/ecs#>). For example, project name, person URI, group name, theme. For people (affiliates) all authors or co-authors with an ECS URI (containing string “ecs.soton.ac.uk”) are listed, together with the documents they authored or co-authored. As already noted the entity name or value might contain words which do not really contribute to the classification and as such there may result in a mistake being made in classifying the document. This is because of a diverse use of terms between the user-defined metadata (documents’) and those from a vocabulary-controlled schema (ECS metadata). Similarly these cannot be easily ignored and as they are only recognisable with human intervention an improvement or further work on the system to allow the user to correct the classification is desirable.
3. In order to prepare the data for comparison of attribute values to decide which items are similar some “data cleaning” is essential to remove non-significant words and characters. As the attribute values and the full filename are divided into words white space (%20 in RDF data), punctuation, connecting words like “is”, “and”, “of” are removed. Like indexing, the attribute values would benefit more if words were selected in a careful and more informed manner (such as concepts from a thesaurus ([Tudhope and Binding, 2008](#))), which unfortunately cannot be done easily automatically in this case. The words (from both the attributes and entities) are changed into lowercase to cater for differences in capitalisation between the system and individuals. Leading and trailing white space is also removed from words.
4. Considering the emphasis in research of the usefulness of the directory hierarchy, it is treated as a source of metadata, with the folder and filenames being used as

keywords for the document in question. These, like other attribute values, are also compared with the entity names or values and the document is listed as associated with the entity if there is a match (partial or full).

An example of viewing people in ECS who have documents related to them and the type of relation is shown in figure 5.18.

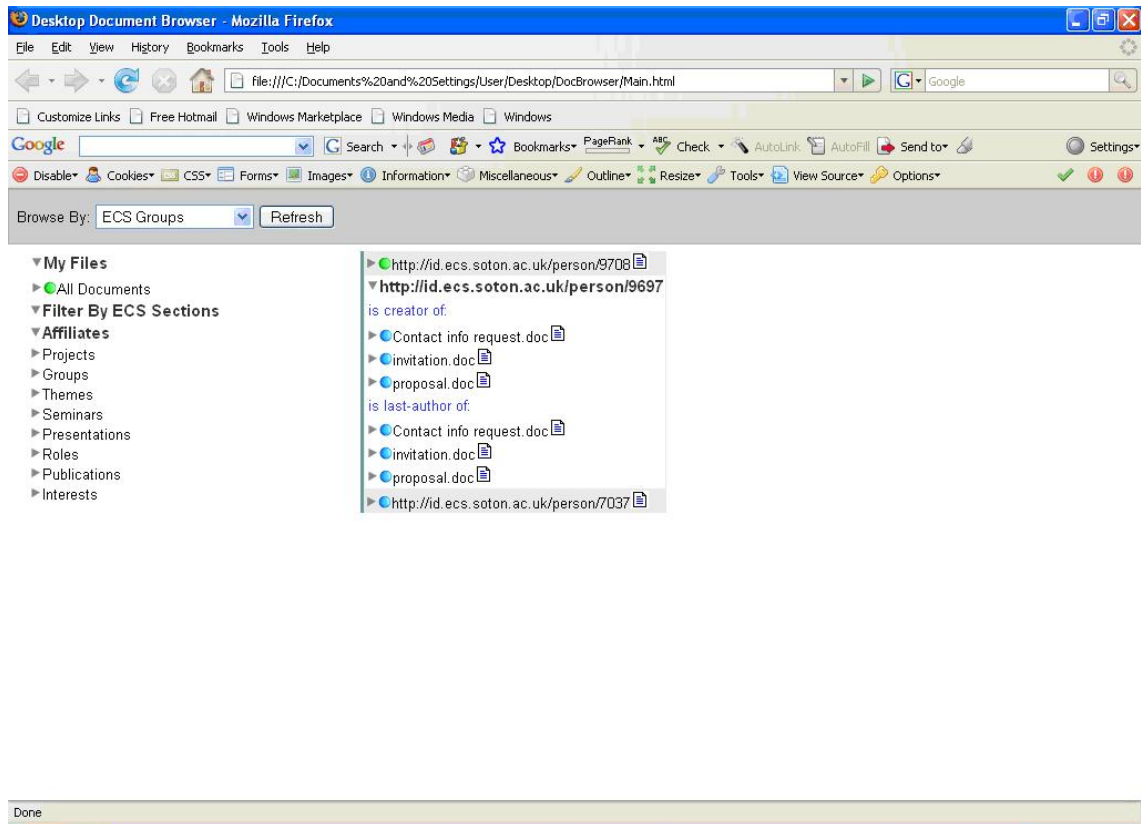


FIGURE 5.18: Filtering documents by ECS affiliates

The user interface for browsing document metadata is presented as a two-part window similar to the file/folder explorer mode of the Windows user interface with whole entities shown on the left-hand side and details shown on the right-hand side.

5.4.8 Discussion and Challenges

5.4.8.1 The Document Browser Architecture

The document browser demonstrates how to utilise document metadata to provide an associative browsing interface visualising relations both within the desktop and with other relevant semantic datasets. A RDF parser is an essential component of the program and the Tabulator open-source AJAR library provided that function. The Tabulator outliner interface also proved useful in visualising predicates relating the different entities

used in the program and in helping to understand RDF data. The query-by-example method is seen as an important function in browsing, and instead of extracting qualifying entities to browse in a different view as in Tabulator, these are presented *in situ*.

To sum up metadata is extracted from Microsoft Office documents, converted into RDF format, then parsed and queried for comparison and integration purposes. The results are then visualised in a web browser in outline format. Figure 5.19 shows the overall architecture of the system.

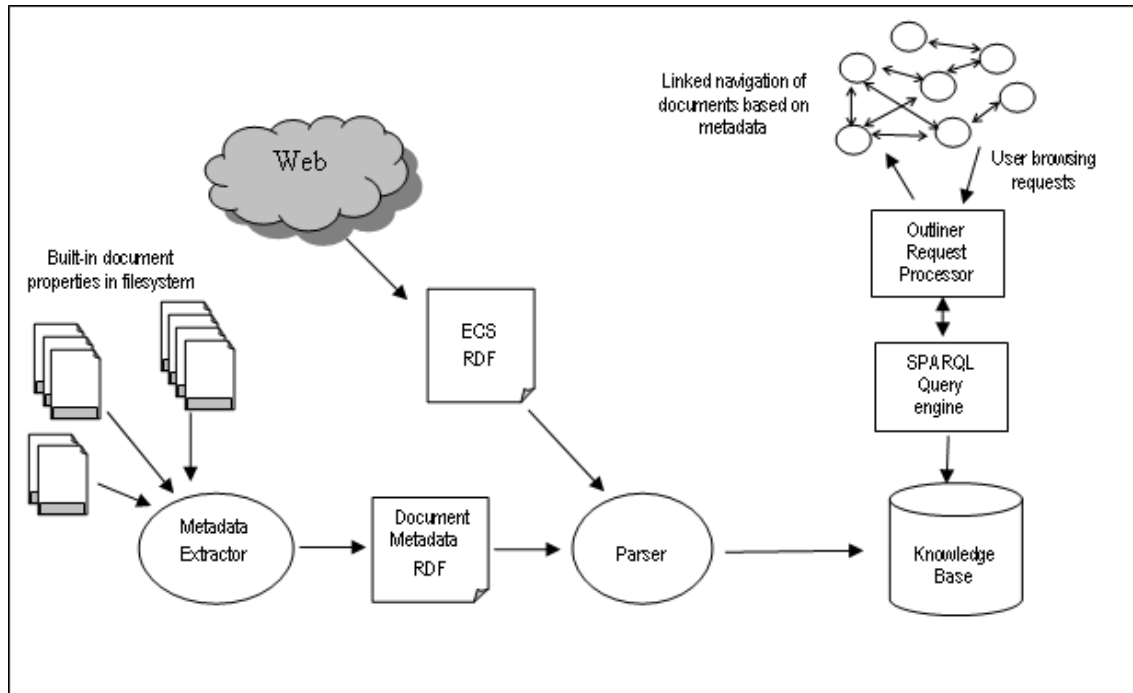


FIGURE 5.19: The overall system architecture

5.4.8.2 Determining Similarity

Simple comparison methods utilising queries have been used so far to determine if resources are related. This solution works well for single-valued or pre-ordered (for example, time and well-ordered sets of terms) metadata. For multi-worded file and folder names and property values, however, this method does not work very well. It is difficult to decide automatically whether to treat phrases as a single string to compare against or not such that related objects are retrieved, as this may be domain-specific knowledge that requires human interpretation. There is a need to use some intelligence in comparing these multi-valued metadata. Information retrieval methods have tended to index documents on their subject after analysis of their constituent concepts. The terms are then combined based on search terms provided by the user during the search process and this is known as post-coordinate indexing (Will, 2004). Post-coordinate indexing is helpful for conducting specific searches, where the searcher can specify the terms that

could be used by the system to derive many possible combinations. The comparison of multi-valued metadata in this instance can also take the form of logic and inference rules if the comparison is based on the semantics of the data.

The system also does not cater for synonyms (different words with the same meaning), polysemy (words having multiple meanings), plurals (apart from the ones that are made of the singular plus an “s”).

While browsing, the current system is only able to query and present results once, querying for similarities from the set of results uses all the metadata again. Progressive refinement of results will help users reduce the amount of results and hence aid in better organization and reminding. In addition query results presentation inside the browser interface result in expansion of the tree which can result in the user being “lost” during browsing.

The example in figure 5.20 shows how the browser retrieves the wrong documents based on associating using single words instead of the whole phrase. These were arrived at as a result of using the individual terms “personal”, “information” and “management”.

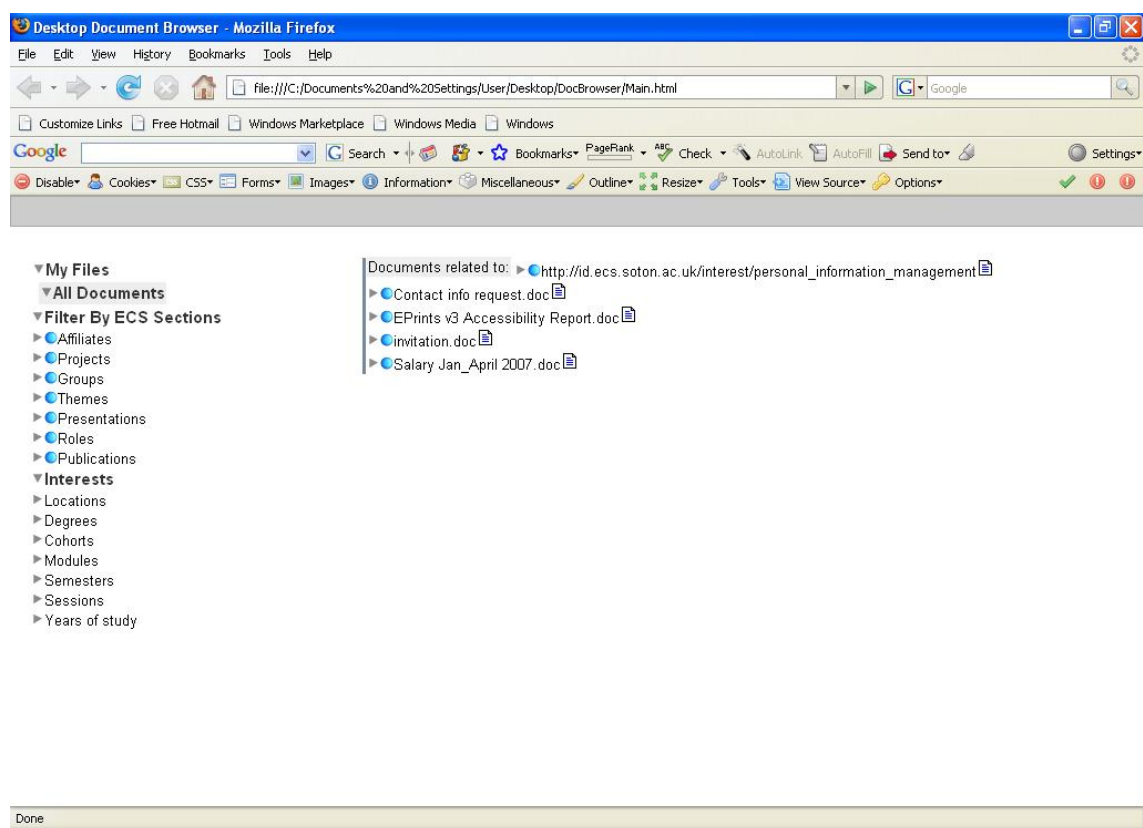


FIGURE 5.20: Wrong classification as a result of using individual words

5.4.8.3 Working with RDF Data

The physical separation of RDF web data also posed a challenge while using it with the RDFParser. Unlike a database where all the records are available in a single location, RDF data is usually kept in separate files which are for specific things. Loading data in the parser only loads the Subjects, Predicates and Objects found in that particular RDF file. URIs encountered as objects in the parsed data must be explicitly loaded for parsing. One of the reasons the parser does not do this is that if the inference engine downloads data for any new URI it comes across, it will continue without limit in an unbounded open web (Berners-Lee et al., 2006).

Web data is needed in the document browser in a manner that allows queries for information to be executed without the delay caused by waiting for URIs to be parsed. For example, when the user requests their documents to be filtered according to publications in ECS, all these publications are not available for immediate access from the knowledge base. Although they can be loaded as RDF Site Summary feeds (RSS 1.0), they are not easy to traverse as they are made of a list or `rss:item` elements which cannot be further loaded easily (see figure 5.21). For example, the node highlighted in figure 5.21 has no URI associated with it. What is needed is therefore a kind of limit set, such that data can be parsed/loaded to a defined point beforehand, so that the process does not interfere with browsing.

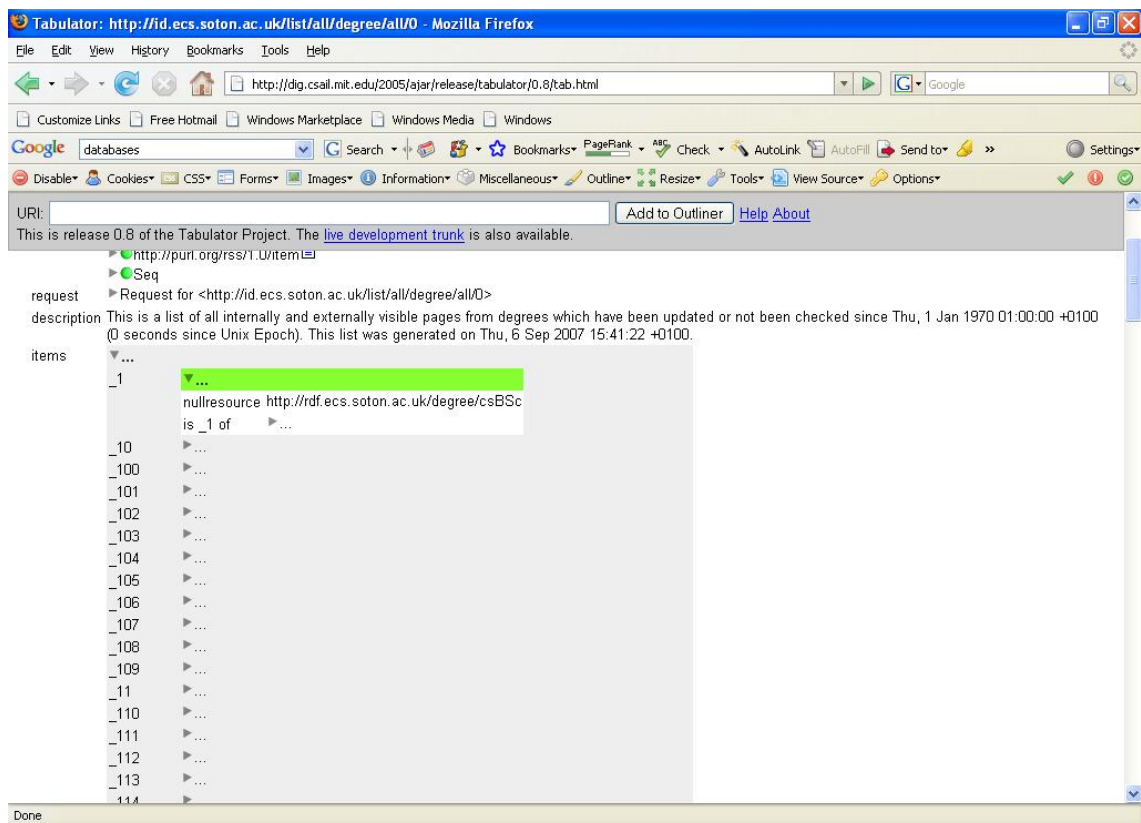


FIGURE 5.21: The structure of a list of all degrees in ECS, visualised using Tabulator

The parser is also slow, having to automatically load and parse ontologies it comes across in addition to the data at the URI loaded. This problem is less significant if this is done when the application starts, as already stated.

5.4.8.4 Querying During Browsing

The Tabulator RDF parser depends on the user selecting the node (URI) to expand during browsing. The Document Browser needs to access and “expand” some nodes in the background while checking if resources are similar. An efficient method of “expanding” RDF nodes either beforehand or during the browsing task has to be used to prevent delays during browsing. The problem is exacerbated by the fact that user data will increase over time, so the program should be able to cater for large data sets. A problem was also encountered with accessing and expanding RDF blank nodes. In the program URIs are used to identify resources and load them into the parser. It proved difficult to load RDF blank nodes without user intervention.

5.4.8.5 Moving Away from Tree-like Navigation

The author made a decision to abandon the tree-like navigation interface for a number of reasons. Firstly, expansion of nodes resulted in extension of the viewable screen space, thereby obscuring the overview of the whole structure. This has been previously recognised in tree-like visualisations. Secondly, though the structure could depict associations between documents based on the file attributes, it is not optimal for knowledge browsing. That is, it could not be easily used to present the document set in a way that could expose the concepts and knowledge embedded within and allow browsing of these without digging through the (possible very long) file list first. Thirdly, associations could not be easily computed and shown based on similar concepts if they are contained in different attribute-value (predicate-object) pairs. Even for the same attribute-value pairs, values could not be easily matched or compared without further processing and analysis (partial matches possible especially between multi-valued object values).

The Tabulator’s outliner interface had already been accessed by users and criticised for failing to provide a cognitive map ([Berners-Lee et al., 2006](#)) and since our objective was to aid users recall and retrieve information by reducing their cognitive load this was viewed as not being an optimal path to achieving our objective.

5.5 Summary

Experiments conducted in a step-by-step effort to solve this problem have been discussed. The challenges encountered as a result of the architecture of the data and limitations

on processing were also discussed. The reasons for a change of idea with regard to the approach taken for the solution especially with the benefit of the approach to the user in browsing and discovery of documents were also presented. The following chapter presents conceptualization and implementation of a different method for effective retrieval and discovery of documents from hierarchies other than the simple point-to-point following of links already presented.

Chapter 6

The *Desktop File Browser* (DeFiBro)

6.1 Introduction

This chapter describes the approach taken to build and demonstrate a method of supporting retrieval, associations and discovery of personal documents organized using the hierarchical file system structure. The underlying organization model for exploration of documents through their metadata is presented. This model is hypothesized to improve accessibility of documents through exposure of the metadata to provide shorter and precise paths to documents. The method used to process the metadata for presentation and linking of documents through the interface is explained and the overall system architecture is presented.

The semantic data gathered before is acted upon by several modules based on this model to provide what is posited to be an intuitive interface to the hierarchy that exposes the documents better for easier browsing, finding, reminding and discovery. This forms the first part of the implementation work. The interface implemented, the *Desktop File Browser* (DeFiBro) offers a browsing interface for retrieval of desktop documents.

The second part of this chapter presents an experiment undertaken as an example of how the documents and their metadata could be bundled and exported from the interface to organized stores that require users to specify metadata while depositing documents from the desktop. In general, bundling documents and their metadata in a common and shareable way could help in access and transfer of documents between different stores and devices given a protocol for such.

6.2 A Model for Desktop Document Retrieval and Association

A lightweight model that utilizes the existing document attributes is proposed to facilitate retrieval and discovery of documents from file system hierarchies. The model is based on the metadata derived from the file system, using Semantic Web concepts for structuring and processing the metadata and determination of similarity between documents.

Document attributes including filenames and the hierarchy structure provide a set of user-defined concepts that were used in organizing documents and are more likely to match the user's mental model and also more likely to be effective reminders. Also the folder-hierarchy already implies some clustering but "hides" documents inside the nested clusters. These and other document attributes available as file properties in the file system are gathered as attribute-value pairs by a file system crawler module.

As already stated the Semantic Web provides a common way of integrating and sharing the information by exposing the semantics. The document attributes are structured according to an ontology and stored in RDF form. Semantic data also provides implicit *facets* which can be used as simple dimensions for navigation and provision of context. In preparation for processing of RDF data, the semantic metadata is parsed and queried using SPARQL modules. Use of a parser instead of a triple store implementation (which typically includes a query engine and an inference engine on top of a database engine) is adopted to allow RDF processing on the client, thereby minimizing overheads associated with database storage and processing.

6.2.1 Metadata Processing

6.2.1.1 Building the Document Metadata index

In hypertext and hyperindexing indexes serve as aids that facilitate the location of objects ([Bruza and van der Weide, 1990](#)). If ordered, index entries can facilitate the quick location of relevant entries.

The semantic metadata is processed to build for each document a forward index by deriving terms from the metadata values which then serve as keywords in document recognition, retrieval and clustering. The terms are actually any text fragments which are recognised by the algorithm as single words but might only make sense to the creator as they may be "codes" used to describe the files and folders. The algorithm for building the index is as follows.

For each document:

- First the metadata is queried for all property values pertaining to a given document. The extracted property values are all treated the same irrespective of property value. While being extracted some pre-processing is carried out to select useful terms in the multi-valued values. This is done by removing stopwords (for example words like “and”, “the”, “is”), punctuation (replace with spaces to separate terms) and converting all the values to low case to facilitate easy comparison. Stemming, which is usually done in indexing to eliminate variation caused by presence of different grammatical forms of the same word (for example, “research” and “researched”), is not carried to avoid intervening with user’s intended meaning of concepts.
- The string value is then divided into terms according to spaces.
- The derived terms are stored as a document index set: Doc Uri \rightarrow t1, t2, ...tn.

6.2.1.2 Buiding a Unified Metadata Index

All the documents’ forward indexes are then used to compile a unified metadata index for the whole document set. The indexes are further enhanced by lookup, selection and addition of synonyms from WordNet. WordNet is a large-scale English lexical database developed at Princeton University (Miller et al., 1990). It is organized into syntactic categories such as noun, verb and adjective. Collection of words that have similar meaning and underlie a lexical concept, called synsets, are provided to differentiate word senses from each other. WordNet offers a chance of matching semantically related words through examination of the word senses to determine if concepts defined are the same.

The process of building the indexes is shown in figure 6.1.

6.2.2 Term-based similarity measurement

Similarity serves as an organizing principle by which objects can be classified, concepts formed and generalizations made (Tversky, 1977). A wide variety of access methods in hypermedia and information retrieval use some underlying notion of similarity, which can be based on either exact or imprecise matching of terms (Tudhope and Taylor, 1997). Retrieval systems employ some kind of similarity measurement to determine and present or integrate related items together. Most automatic linking hypermedia applications employ some kind of semantic similarity measure for automatic creation of links (Tudhope and Taylor, 1997). Semantic similarity, a concept where a set of documents or terms within lists of structured terms are assigned a metric according to likeness of their semantic content or meaning, can be used for entities described using a common schema such as an ontology.

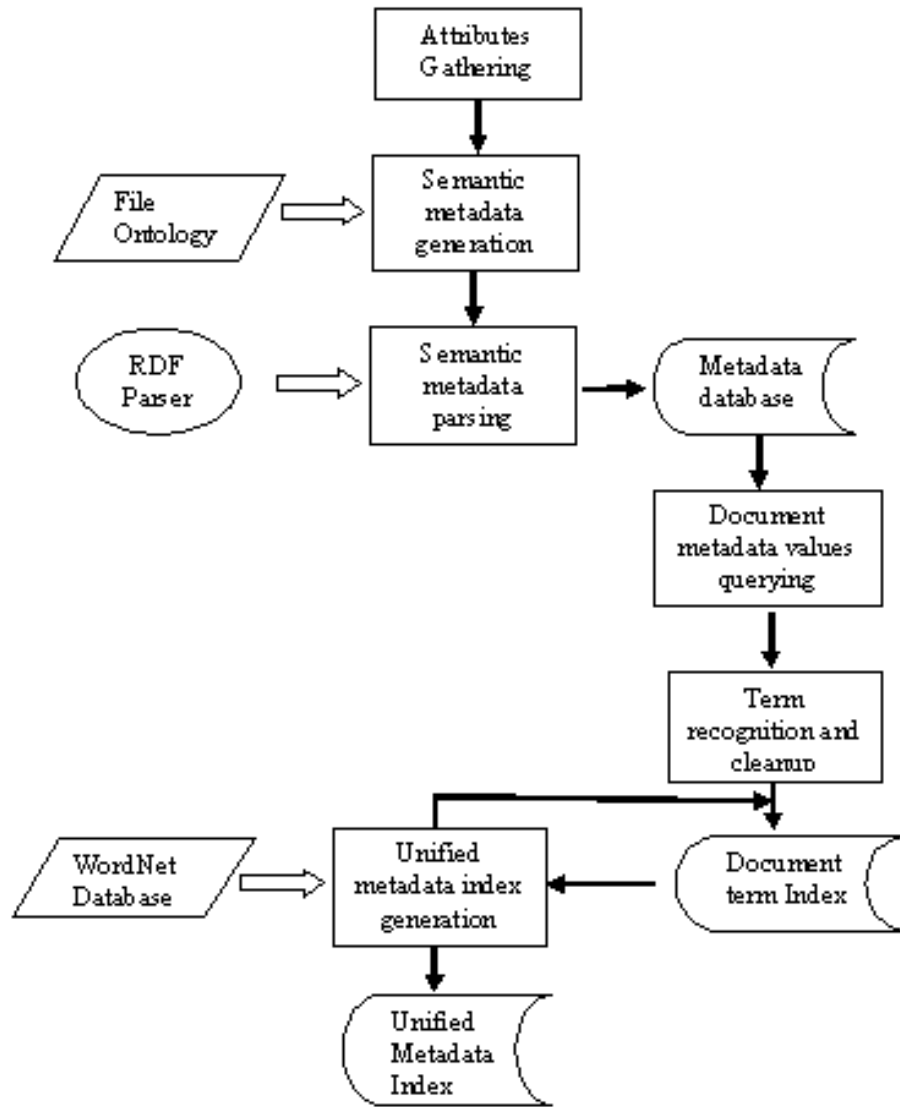


FIGURE 6.1: The index derivation process

Document similarity has been measured especially using the vector space model (Salton et al., 1975), a common geometric method used in data integration, querying and similarity-based retrieval. The method makes use of weighted sets of terms for documents to compute their similarity. The TF-IDF (term frequency-inverse document frequency) weighting schema is widely used to assign the term weights which are then used by the model to compute similarity between two documents.

Using the vector space model, document D_i is represented by a t -dimensional vector

$$D_i = (D_{i1}, D_{i2}, \dots, D_{it})$$

where D_{ij} represents the weight of the j th term. The weight reflects the ability of the term in describing the document and is proportional to the frequency of the term in the document. The weight is also related to the frequency of occurrence of that term in the entire corpus of documents. The inverse document frequency (IDF) is incorporated in the weight to counteract the effect of common terms that frequently occur in the document while boosting the weight of the meaningful terms that occur rarely.

Given the vectors for two documents, a similarity coefficient between them $S(D_i, D_j)$ is computed as either the inner product of the two vectors or an inverse function for the angle between the corresponding vector pairs. Documents are represented as points in a coordinate space such that two documents with similar index terms are represented by points that are very close together in the space. Generally, the distance between two document points in the space is inversely correlated with the similarity between the corresponding vectors.

The model described for documents above involves creating indexes for documents by ‘flattening’ the hierarchy information and terms extracted are non-structured since they have been isolated from their semantic description. Semantic similarity measures based on ontologies such as edge-based and node-based approaches ([Bernstein et al., 2005](#)) cannot therefore be used in this instance. Since the TF-IDF weighting method is based on assigning a weight to a term which is calculated based on the number of times a word appears on the document, and we are focusing on terms derived from the metadata (which are fewer and unlikely to repeat), this method was also abandoned in favour of methods which are based on comparison but not weighting of the terms.

Currently, adaptations of the vector model which employ semantics have started appearing in literature. The semantic vector retrieval model ([Li, 2009](#)), for example, uses extracted concepts based on an ontology instead of keywords extracted from the document to represent the document. The structure of the document is also utilized to determine the weight of concepts and similarity calculations takes into consideration the relations between the terms. Such methods can however only operate in specific areas where a domain ontology has been defined. Personal documents, on the other hand, can be from diverse areas and an ontology may be difficult to come up with except if the documents are assessed first and the ontology developed incrementally.

Because the terms extracted include those from the hierarchy structure where the folder names form a structure showing relations by location between the documents, distance-based semantic relatedness measures ([Cross, 2006](#)) may be applicable, even if only as part of the whole solution. These are based on counting the number of edges (which can also be assigned a weight based on depth or density of a node or the type or strength of a link) between the concepts, with a shorter distance signifying more relatedness. The methods were not considered for this research since they are already covered by considering the commonality of terms in the above approach (documents in the same

directory are more likely to share more terms), and the fact that the mental model behind assignment of documents to folders has not been well studied and is not clearly defined enough to warrant assignment of weights to the terms themselves.

Another measure of similarity that has been used in hypermedia systems is semantic closeness (Cunliffe et al., 1997). Manually created terms that are descriptors of media items like images and text passages and that have semantic relationships between them were used in the semantic hypermedia architecture to provide browsable links and nodes. The semantic information was then used by the system to augment navigation support and retrieval by measuring the semantic closeness of terms provided in a query, which included reasoning over the relationships in the index space and imprecise matching based on transitive semantic relationships. Semantic closeness is however more difficult to apply in our case since there is no structure specifying semantic relationships between the metadata terms derived. The user in this case is in a better position to specify these than trial of automatic means by the system.

Amos Tversky's similarity model (Tversky, 1977) expresses similarity between objects as a feature-matching process; a linear combination of the measures of their common and distinctive features. The feature matching, unlike the vector space model, is directional and asymmetric.

Tversky's contrast model considers a domain of objects under study $\Delta = \{a, b, c, \dots\}$ where each object in Δ is represented by a set of features or attributes; A for a, B for b, C for c, and so on. Regarding these, Tversky noted that it is important to note that the total database concerning a particular object is rich in content and complex in form, and only a limited list of relevant features is compiled to perform the requested task. Also, the term "feature" usually denotes the value of a binary or nominal variable, although it is not restricted to these two and is also applicable to ordinal and cardinal variables (i.e. dimensions).

The contrast model states that given Δ there exists a similarity scale S and a non-negative scale f such that for all a, b, c, d in Δ

$$S(a, b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A), \quad (6.1)$$

for some $\theta, \alpha, \beta \geq 0$

With α and β being the weights assigned to the distinctive features of a and b respectively. When $\alpha = \beta = 0$ we are only interested in that object a and b share some features, therefore the more features the two objects have in common, the more similar they are. The model can therefore be adapted not to include distinctive features in this way. This serves our purpose well since the distinctive terms in documents may differ in number and do not say a lot about how the documents are different from each other.

A similar approach is therefore adopted for assessing the relatedness between two documents by looking at their similarity as a function of the number of terms shared while ignoring the distinctive terms as they serve no purpose in the comparison. This effectively selects a cluster of documents according to the number of similar terms shared between them. The terms representing a document are the keywords derived from the metadata values including the filename and folder hierarchy, plus the synonyms derived through WordNet that exist in the document index space.

A document-document similarity matrix is therefore built after the index is created based on pair-wise comparisons (likeness rather than equality) of terms in documents' forward indexes. Each row (document) is then dynamically extracted and ordered by cell value on a similarity request (selecting a document, thereby requesting more details) on the interface during browsing.

Given D_x as the list/set of terms for document x , and D_y as those for document y , $S(D_x, D_y)$ represents similarity between document x and document y . The whole matrix is initialized to zero before the comparison starts.

$$D_x = \{Tx_1, Tx_2, \dots, Tx_n\}$$

$$D_y = \{Ty_1, Ty_2, \dots, Ty_m\}$$

$$\forall i, \forall j, \text{ if } Tx_i \text{ like } Ty_j \text{ then } S(D_x, D_y) = S(D_x, D_y) + 1$$

That is,

$$S(D_x, D_y) = (D_x \cap D_y) = \sum_{i=1}^n \sum_{j=1}^m (Tx_i \approx Ty_j) \quad (6.2)$$

The comparison is described as likeness or approximation of similarity (rather than equality) between terms since partial matching based on stemming and synonymy is taken into account. The partial matching is also a result of the data cleanup that was done during term extraction (removal of punctuation, stopwords and conversion to lowercase). With this approach document D_i is more similar to document D_j than to document D_k if $S(D_i, D_j) > S(D_i, D_k)$, that is, more common terms between D_i and D_j than D_i and D_k .

6.3 Metadata-Based Exploration through an Interface

The model described is then used as an underlying organization to develop an interface for supporting retrieval and discovery of documents. The following are a description of important aspects of the interface.

- Overview - provision of some sort of overview over a document collection is essential as discussed previously. Facets have already been seen to provide good

overviews ([Kobilarov and Dickinson, 2008](#); [Henderson, 2005](#)). Traditionally, filing systems such as sorting by alphabetic order and divisions into categories or by chronological order have been used for classification and organization so that access later is easy and efficient. The interface implemented provides two kinds of overviews: an alphabetic and numeric index of document names, and overlapping facets in the form of authors, organisations and keywords (terms) derived from all the metadata values. These provide numerous routes to a desired document depending on features mostly associated with the document. Alphabetic indexes are a common method used by most applications to provide quick exposure and access to data items based on a selected property. There are even applied in a limited manner in operating system hierarchy visualizations inside specific views, for example within a folder, to provide quick navigation from item to item.

- Context - There is a challenge of providing the user with context as well as detail ([Faichney and Gonzalez, 2001](#)) as in the tools discussed in chapter 2. Initially the context of documents in the filesystem was the hierarchy structure itself. The folder structure have been 'flattened', the layout and positioning of documents in the structure has been relinquished in favour of a flatter and more visible linked keyword structure. Within the facets provided the user can select items of interest to view clusters of documents based on the facet values. Context for a selected document is provided by related details consisting of linked metadata terms (to get the documents to select one another based on similarity of terms) and similar (associated) documents to the selection. In addition to providing context, linked metadata and similar documents add additional dimensions for navigation within the facet-based overview.
- Details - File thumbnail representations have been found to be effective for locating and organizing documents in user interface studies dealing with documents ([Faichney and Gonzalez, 2001](#); [Lucas and Schneider, 1994](#); [Mander et al., 1992](#); [Robertson et al., 1998](#)). At a glimpse the user can see whether a document is a picture, table or plain text and can quickly decide whether there is need to investigate further by opening the document or whether to check other possible links. They can be especially helpful if the contents are clearly visible and the text is readable as is provided in the implemented interface. A preview of the first page of the document in the form of an enlarged thumbnail is given on hovering the mouse pointer over a listed document thumbnail.
- Extract and Transfer - A 'workspace' panel is provided to enable selection and extraction of desired documents with their metadata. The feature is implemented as a solution to user needs involving difficulty of working with documents across several folders and to address the need for creating desired groupings for immediate work purposes that might not be satisfied by the similarity clustering implemented. The importance of availing and packaging semantic document metadata

with documents is also highlighted in tasks involving transfer of documents from the desktop to other organized document stores. An experiment with this aspect is discussed in section 6.5.

The comprehensive interface enables exploration of all documents in the filesystem by link-guided browsing through provision of linked metadata, providing a significant shift from structure-guided navigation offered by hierarchy traversal methods to knowledge-based browsing. The design and production of system offering the described interface is described in detail in the following sections.

6.4 Improving File Browsing on the Desktop

6.4.1 Processing the Semantic Metadata for Presentation on Interface

6.4.1.1 Parsing RDF Data

To be able to work with the semantic metadata stored in the RDF file there is need to process the RDF and store it in a convenient knowledge base to allow the client-side implementation to query the data. The Javascript library implementation of Jim Ley's RDFParser in Tabulator ([Decentralized Information Group, MIT, 2006](#); [Berners-Lee et al., 2006](#)) provides open-source Asynchronous Javascript and RDF (AJAR) scripts which can be utilized to work with Hypertext Transfer Protocol (HTTP), XML, RDF and SPARQL Protocol and RDF Query Language (SPARQL). SPARQL is a powerful query language for access of RDF data. Initially interest was taken on the Tabulator as an *RDF browser* allowing visualization and interaction of the metadata collection as discussed in chapter 5.

The library was downloaded and utilized to parse the RDF files and store them as triples of the form subject, predicate and object in an in-memory database. The RDF triples can then be accessed using SPARQL queries. Just like in Tabulator a web browser is used for browsing of the metadata utilizing the HTML Document Object Model (DOM) to work with objects on the web page.

By running on the user's machine to enable it to perform linked data protocols ([Berners-Lee et al., 2006](#)), the library suited our purpose of creating a simple and easily portable personal application which deals with confidential documents from the user's desktop. Also no installation of programs and database backends is required, hence making the code easy to adopt for experiment purposes.

6.4.1.2 Facets for Browsing

The metadata itself already presents useful facets which can be used to browse the documents. Common facets provided in systems for browsing documents include author, time (created, modified), name, size and type. Facets are based on classifying based on properties without analysis of the property values, that is, the values are presented as they are. Facets are rather rigid as they are "mutually exclusive". Providing these on the whole set of documents provides an important view of the document set but does not provide a deeper, across-facets analysis, comparison and filtering.

The implementation therefore provides three of the common facets to browse documents (Author, Organization and Name) and augments these with values from other attributes. Author, title and date have been identified before and recommended for automatic classification of documents by [Malone \(1983\)](#). Other facets recommended include person, topic, time and filetype ([Henderson, 2005](#)). Operating systems also allow re-ordering of search results by file properties like filename, time and author. [Oren \(2006\)](#) also recognised the facets what, who, when, where, and why. as being important for recalling and remembering information. Author and name were therefore selected in this instance for these reasons, organization because of the availability of its values and effectiveness in reminding users about the origin of documents. Time was not used as it had been extensively investigated in other systems before (for example Lifestreams ([Freeman and Gelernter, 1996](#)) and Timescape ([Rekimoto, 1999](#))). However, further study about the most effective facets to use is needed to justify their addition to the interface.

6.4.1.3 Terms and Filename Indexes

The interface also provides a list of "keywords" - browsing terms that are "mined" from the facet values provided and all the other attribute values. An index based on the first letter in a filename is also provided, enabling alphabetic-order browsing of documents by filenames.

6.4.1.4 Implementing the Conceptual Model for Browsing Documents

A conceptual representation of each document is built using its metadata values when the application is initiated. The attribute values, including the folder hierarchy information (from the full filename), are divided into terms, analysed and sorted to form new facet values in addition to the easily recognisable ones that are already implicit in the metadata.

Since most of these terms are natural language, having been specified by end-users, this necessitates some natural language processing (NLP) to be undertaken to prepare for

program analysis. Firstly all the terms are converted into lower-case to make comparison free of capitalization consideration and therefore easier. Then follows the recognition and removal of stop words, recognition of dates and people's names (using day, month, year and names already recognised from other properties like date-modified, date-created, author and last-author as gazetteers respectively) and division of terms based on capitalization, spacing and punctuation. Terms that are found in the gazetteers are added to the appropriate facets while those that are not are treated as other terms.

Words with the same meaning as a term, known as synonyms, are looked up in the WordNet database, and added to the set of keywords for that document. The WordNet synonyms are accessed through the Java API for WordNet Searching (JAWS) ([Spell, 2008](#)). These are used to supplement the keywords such that they can be used to compare and cluster documents for organization and relating in the interface. Since there can be many synonyms for a given term and some may be irrelevant or may not necessarily be useful for the user in relating documents the set is reduced to include only those that are already in other document's metadata. The program however does not try to disambiguate terms and pick up synonyms based on the meaning of the term. For this to be done further natural language processing (NLP) and analysis on the data based on the context in which the term is used will be required. For example the synset for the word "research" is as follows.

research - {research, inquiry, enquiry, search, explore}

based on both the noun and verb senses of the word. If any of the four synonyms occurs in any file's index the term 'index' is then added to that index to signify a relation between the two.

As the terms are being extracted they are added to a structure representing the whole document set without duplicates. This is presented in the interface as keywords for the user to dynamically browse through and select to display associated documents (figure 6.2). The extracted terms for each document form a set which is accessed against other document sets to determine similarity between documents. This similarity assessment compares individual terms by checking full and partial matches. This is done during browsing when the document is selected for display as shown in figure 6.3. The related documents are ordered according to the scale of terms that are similar to that of the selected document.

6.4.1.5 Preview Generation

The metadata collecting module described in the previous chapter also creates previews as the hard drive is being crawled. For the document types concerned (Excel, Word, Powerpoint and PDF) an instance object of the application is created and the document opened in the instance. The first page of the document is then copied into the clipboard

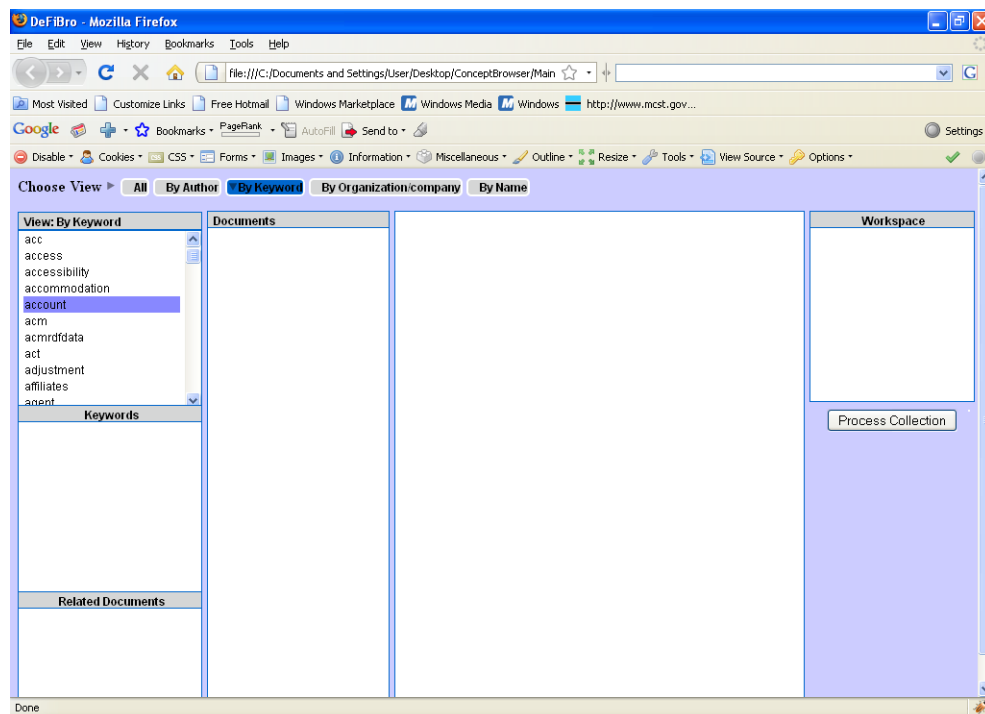


FIGURE 6.2: Browsing by keywords extracted from document set

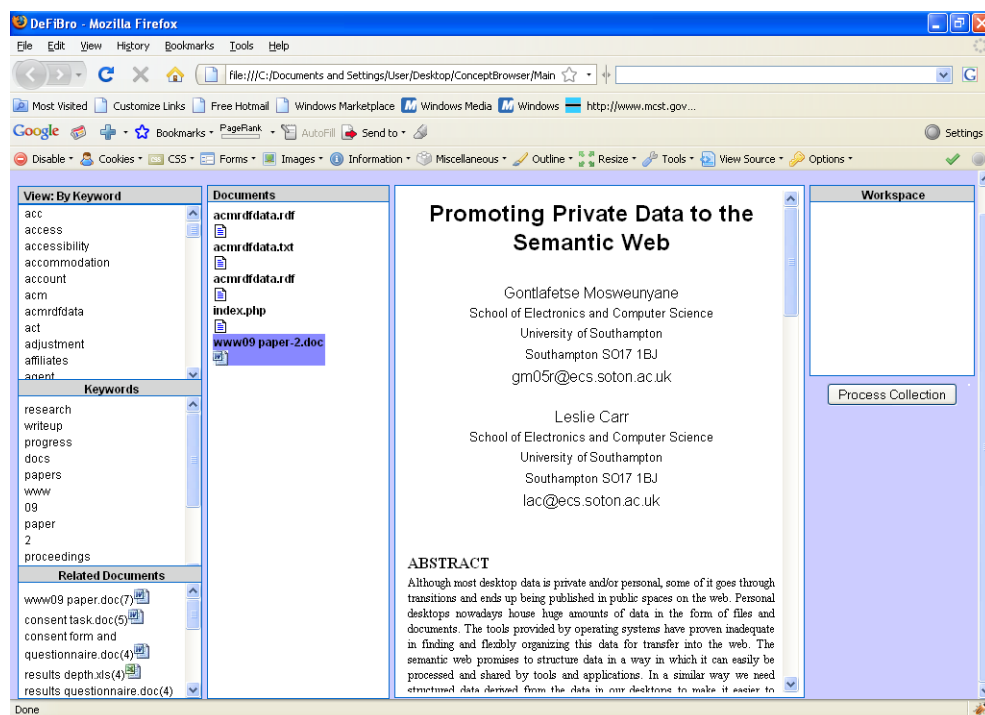


FIGURE 6.3: Selecting a document shows its keywords, preview and related documents

object. The application (together with the original unchanged file) is then closed and the page pasted into a Word Object Instance. The page is then saved as an hypertext HTML document for easy visualization from the web browser interface. A trial was made with other easily-viewable formats (.png, .jpg, .bmp) and this were found to be of a much bigger size which could create space problems on the user's machine.

A folder is created at the same location as the program folders for storing the previews, which are retrieved on a per need basis while the user is browsing the documents from the interface.

6.4.1.6 Challenges in Building the Model

A challenge with the approach described for extracting concepts is that though it is effective in recognising most terms it was observed that it may miss out crucial information when the terms are separated or extract terms that are not sensible due to the user's style of naming of the files, folders and use (or lack) of punctuation. This includes spelling variations by the user. Although gazetteers could be provided for attributes already seen in facets implicit in the metadata there is no way that could be used to "clean up" and recognise all terms in the area of documents metadata since they are highly personal and varied, and therefore cannot be associated with any particular clearly-defined domain.

Similarity depends on context and frame of reference or feature space ([Tversky, 1977](#)). For this project similarity in documents depends on the syntatic likeness of terms in the metadata rather than their semantics, save for the use of synonyms derived through WordNet. Some of these terms have no dictionary meaning, for example names, abbreviations, combinations of numbers and letters and stemmed words, and as such an effective comparison cannot be made between these and other "proper" terms. As such similarity between documents in this context depends heavily on the definition of the structure (hierarchy), file names and any attributes by the user.

6.4.2 DeFiBro Architecture

The system, DeFiBro, has a separate VBScript crawler module which traverses the file directory structure, collecting in-built file metadata, converting it to semantic data and storing it in an RDF file. At the same times first-page previews are generated and stored for later use. The semantic data is the parsed and accessed through SPARQL queries. Before the data is presented for browsing in the interface it is subjected to processing and indexing to enhance it and provide users with more attributes that expose documents and to provide more effective filters. A preview handler module links the document to its preview on the browser.

The overall system architecture is shown in figure 6.4.

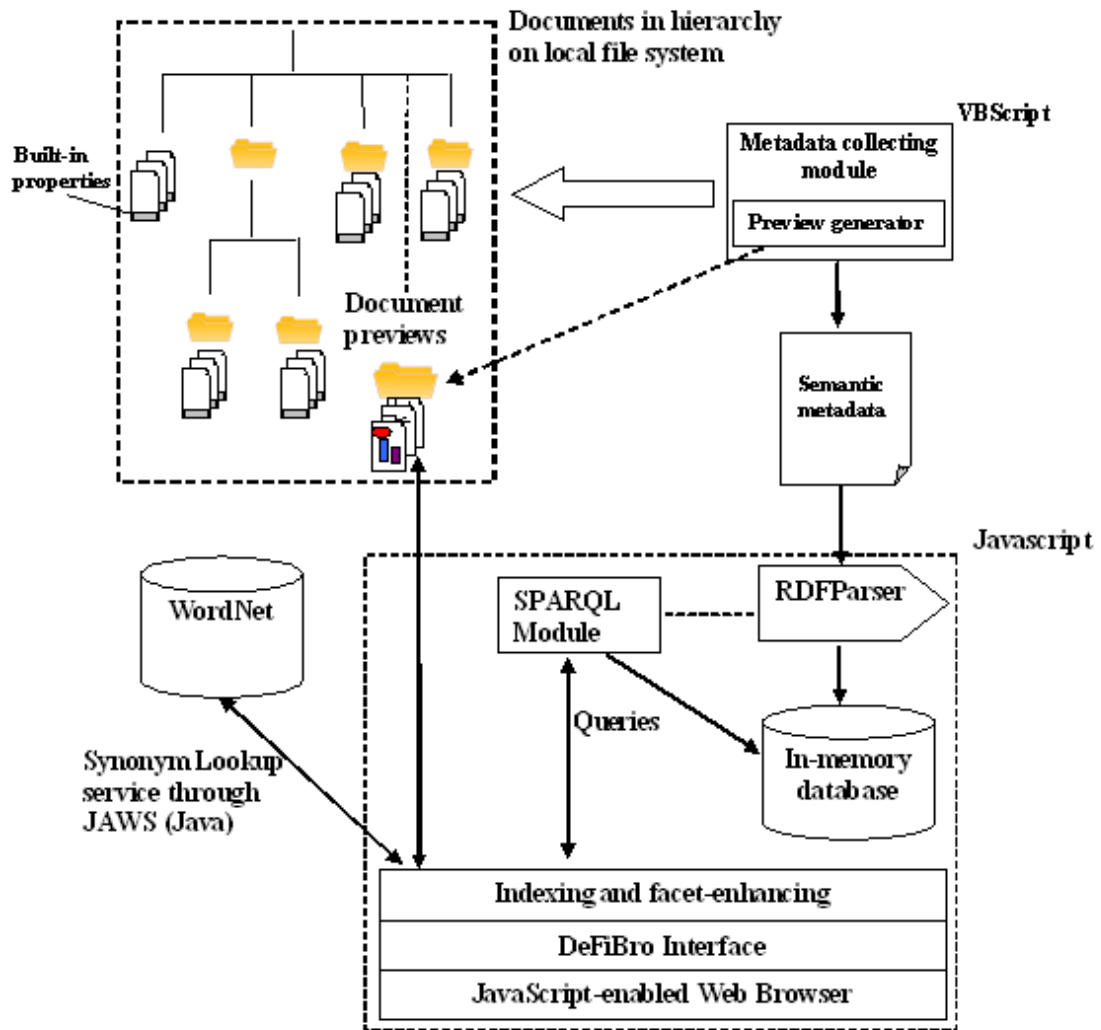


FIGURE 6.4: Overall Architecture of the System

6.4.3 The Browsing Interface

The browsing interface is divided into 6 panels as shown in figure 6.5 in addition to a view selection menu at the top of the page.

The user selects a facet to browse (Author, Keyword, Organisation/Company) and the values of the selected facet are shown in top-most left-hand side panel, figure 6.6.

When an item in this panel is selected a list of documents matching the criteria satisfied by the selected value is provided in the *Documents* panel, figure 6.7.

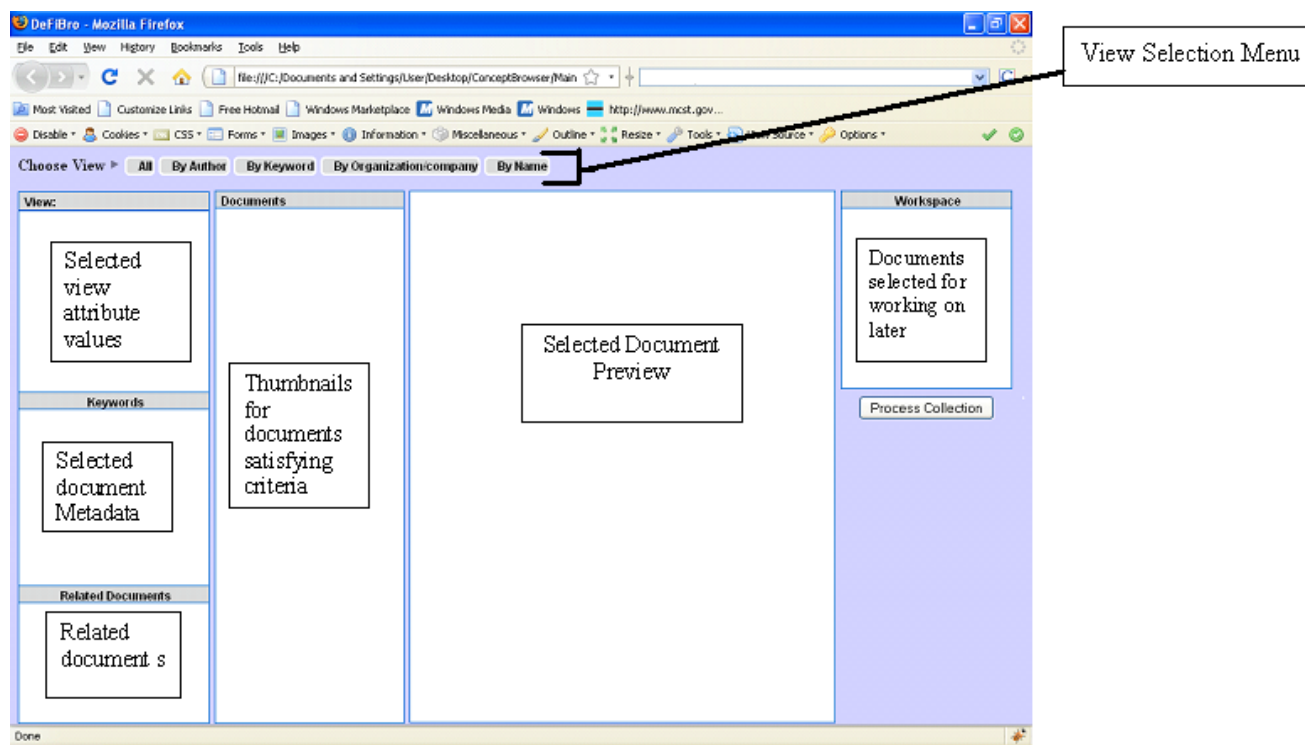


FIGURE 6.5: Browsing Interface Architecture

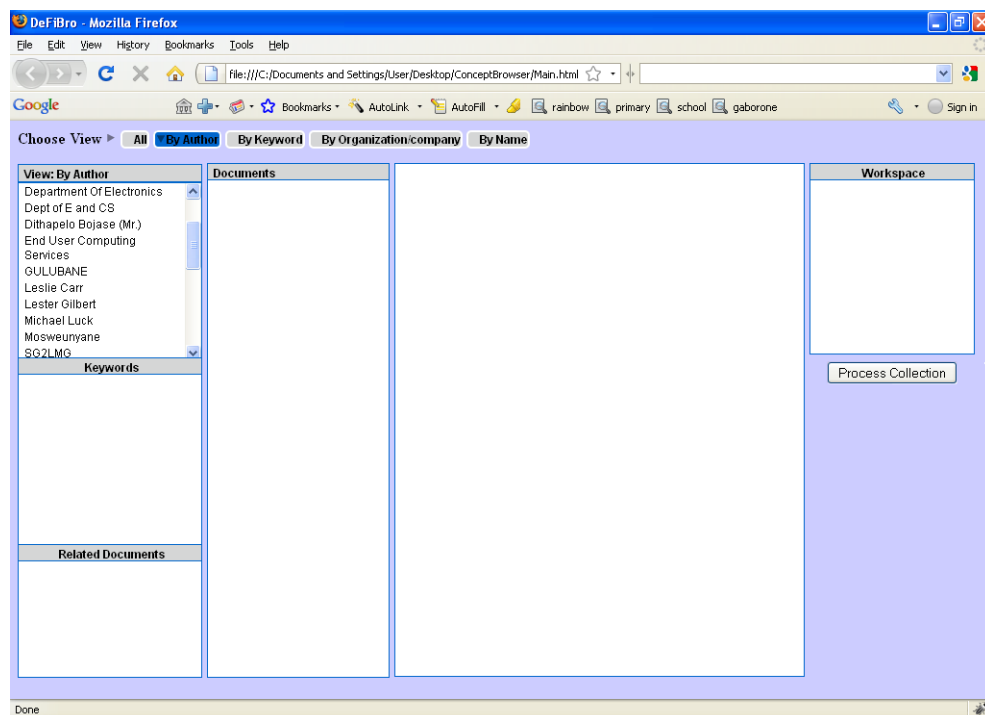


FIGURE 6.6: Selecting the facet 'Author' shows the values of the facet

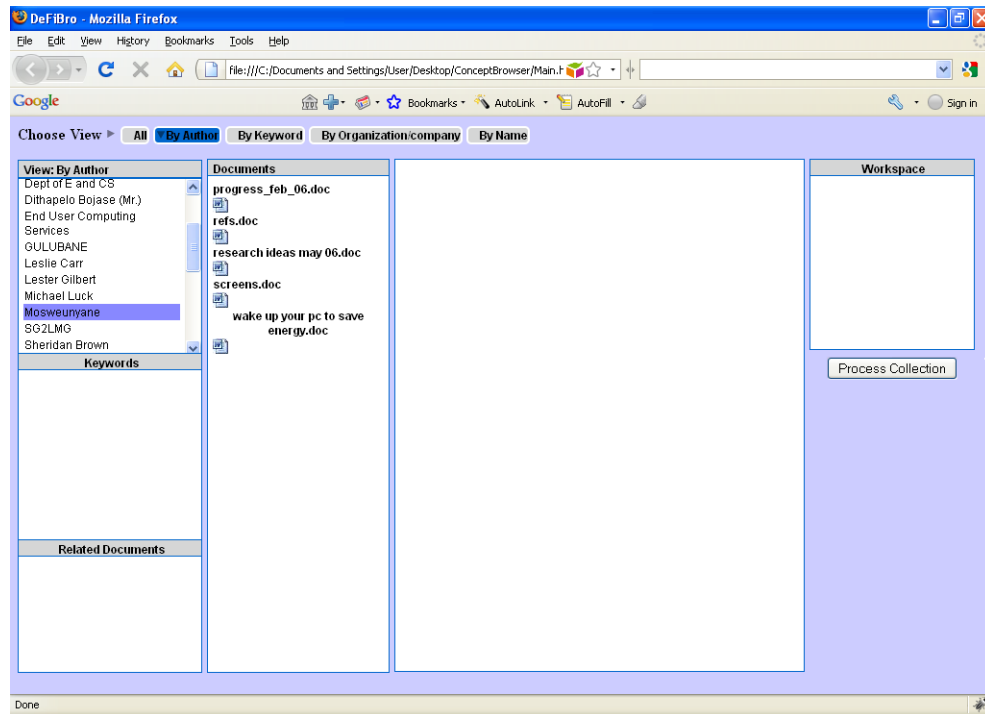


FIGURE 6.7: Selecting the facet value shows the documents matching the criteria

Browsing by filename presents a submenu where the first character of the file name can be selected (figure 6.8) next to the view selection menu) and the corresponding filenames shown in the Documents panel.

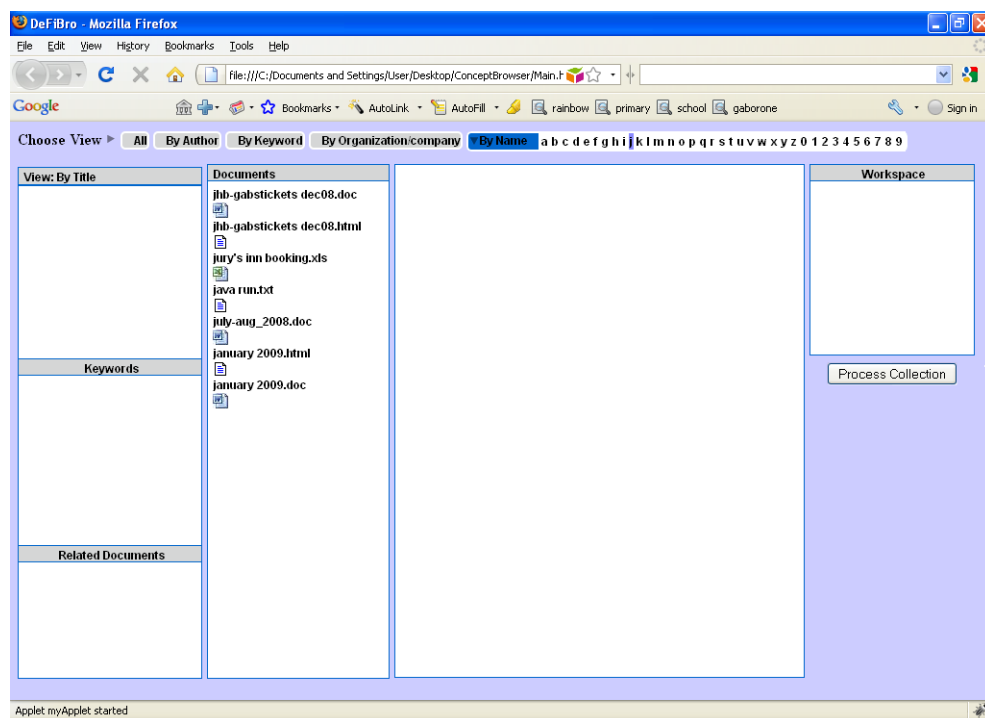


FIGURE 6.8: Selecting the documents based on first letter of filename (filename index)

Pointing to a document name or associated icon reveals its preview in the *preview* panel (figure 6.9, and selecting it reveals the keywords extracted from its metadata in the *keywords* panel and related documents are shown in the panel below that (figure 6.10). These keywords are linked, that is, one can select the keyword to display documents whose criteria it satisfies.

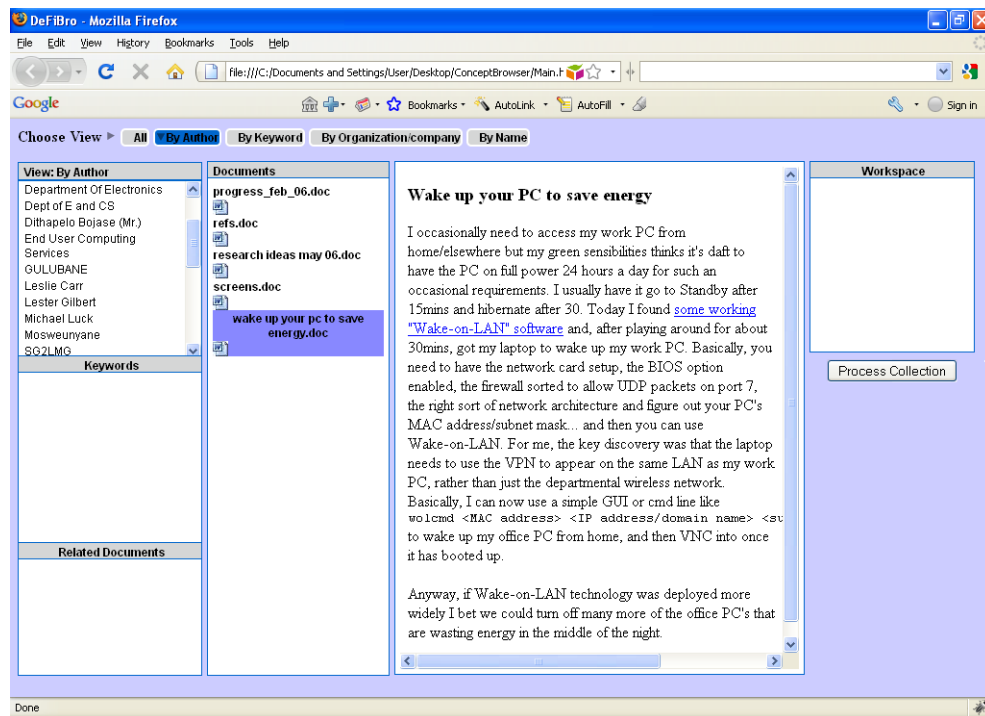


FIGURE 6.9: Pointing to a document name/icon reveals its preview in the preview pane

While browsing the user can collect documents of interest into a collection “basket”, the *workspace* panel (figure 6.11). Documents in this panel can be previewed and selected just as documents in the *Documents* and *Related documents* panels. Documents can also be opened in their respective applications by simply double-clicking on the document as in the desktop.

6.5 Integrating the Desktop with Repositories

6.5.1 Introduction

A repository is an institutional scale collection which feeds off individuals’ document collections and is organized in a specific way. The desktop, on the other hand, represents an individual’s “collection” and “organization”. There is an increasing need for the desktop, as an information source and embodiment of researchers’ workflows and working practices, to be integrated to these other information collections such as the institutional repository (CRIG Unconference, Birkbeck College, 2007). With the availability of a

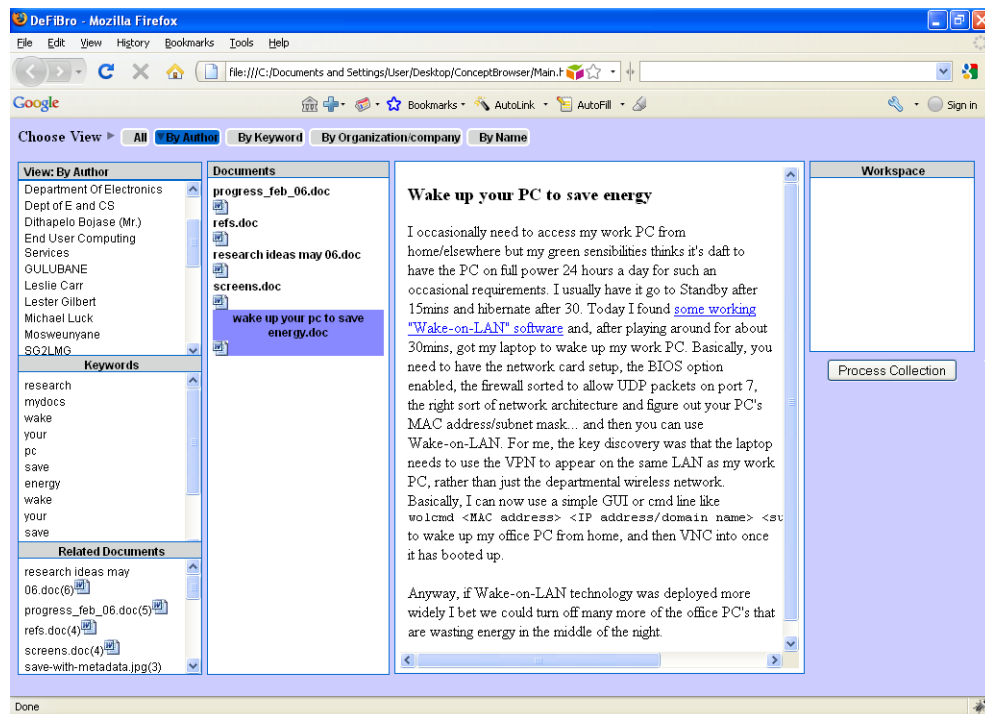


FIGURE 6.10: Selecting a document reveals the associated terms (keywords) and related documents

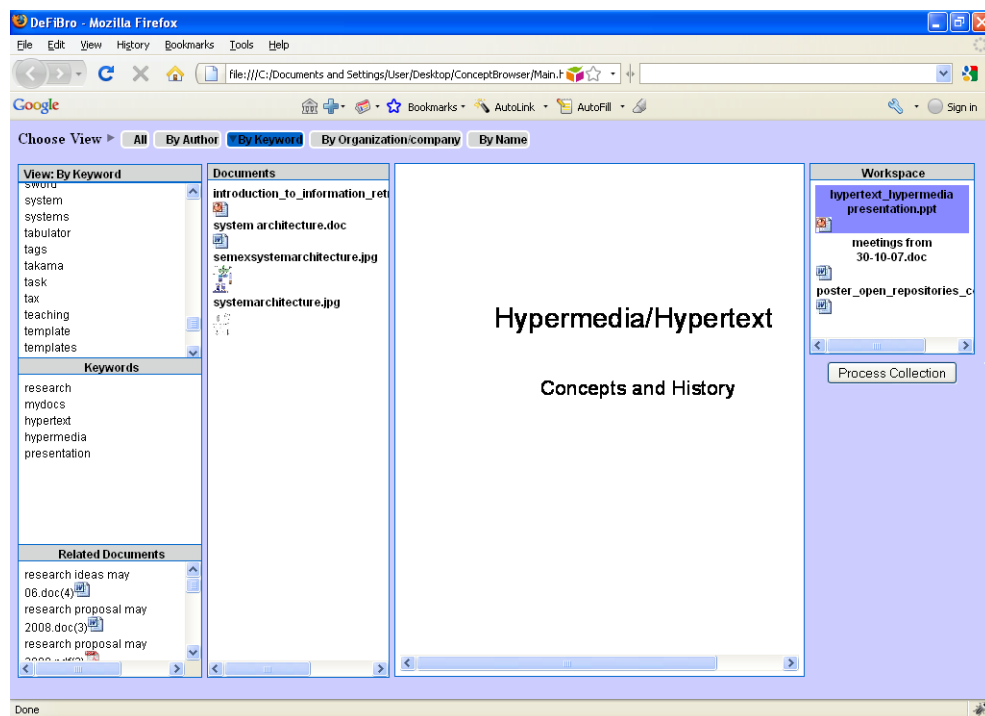


FIGURE 6.11: Documents collected in the “basket”

readily shareable format (in this case the Semantic Web format - RDF and XML) and a protocol connecting the two environments this can be possible. Frequent users of a repository will therefore benefit from an organization mechanism for the desktop that will help integrate desktop data with data from the repository.

SWORD is a deposit protocol which is a profile of the Atom Publishing Protocol (APP), a protocol for publishing and editing web resources (Allinson et al., 2008), which is used in blog websites. It can be used to create SWORD-compliant deposit clients or SWORD interfaces into repositories. The Atom Syndication Format (ATOM⁸) is an XML-based document format that describes lists of related information known as *feeds*. *Feeds* are composed of a number of items (*entries*), each with an extensible set of attached metadata.

The School of Electronics and Computer Science (ECS) at the University of Southampton has implemented a repository for storing publication data called EPrints. EPrints is SWORD-compliant, allowing metadata to be posted with the documents embedded in a file. The metadata has to be in the Eprints XML format.

6.5.2 SWORD Deposits from the Desktop to EPrints

An opportunity to demonstrate how the implementation can also be utilized for integrating the desktop with an organizational document stored was seized given the availability and proximity of these resources to the author. The experiments were carried out using an EPrints SWORD test server instead of the live EPrints repository and involving transfer of single and multiple documents to the repository. Switching to use of the live repository, however, follows the same procedure save for changing the server name to that of the live repository instead of the test one and the login credentials (username) to that of a specific repository user.

SWORD⁹ uses the HTTP POST¹⁰ method to transfer data files and packages:

POST to URI of Collection

It also supports the HTTP GET request and ‘X-on-Behalf-Of’ GET for request of the service document and authentication that allows execution of request on behalf of another user. The service document is returned by this request identifying only those collections to which the target user is able to deposit. For example, the service document below shows that the user can deposit to the buffer (items waiting for approval before being shown in the live archive) at

⁸<http://www.ietf.org/rfc/rfc4287.txt> [last accessed 02/10/09]

⁹http://www.ukoln.ac.uk/repositories/digirep/index/SWORD_APP_Profile_0.4 [last accessed 02/10/09]

¹⁰http://www.ukoln.ac.uk/repositories/digirep/index/SWORD_APP_Profile_0.4#9..Creating_and_Editing_Resources [last accessed 02/10/09]

<http://eprints.ecs.soton.ac.uk/sword-app/deposit/buffer>

and to the inbox at

<http://eprints.ecs.soton.ac.uk/sword-app/deposit/inbox>

of the ECS Eprints archive, together with a specification of the content types understood by the server that could be posted.

```
<?xml version="1.0" encoding='utf-8'?>
<service xmlns="http://purl.org/atom/app#" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:atom="http://www.w3.org/2005/Atom" xmlns:sword="http://purl.org/net/sword/">
<workspace><atom:title>ECS EPrints Repository</atom:title>

<collection href="http://eprints.ecs.soton.ac.uk/sword-app/deposit/buffer">
<atom:title>Repository Review</atom:title>
<accept>application/pdf</accept>
<accept>image/jpeg</accept>
<accept>text/xml</accept>
<accept>application/x-zip</accept>
<accept>image/png</accept>
<accept>image/x-png</accept>
<accept>application/xml</accept>
<accept>application/zip</accept><sword:collectionPolicy>
</sword:collectionPolicy><sword:treatment>Deposited items will undergo
the review process. Upon approval, items will appear in the live
repository.</sword:treatment>
<sword:mediation>true</sword:mediation><sword:formatNamespace>JPEG
</sword:formatNamespace>
<sword:formatNamespace>http://www.loc.gov/METS/</sword:formatNamespace>
<sword:formatNamespace>IMS</sword:formatNamespace>
<sword:formatNamespace>http://eprints.org/ep2/data/2.0</sword:formatNamespace>
<sword:formatNamespace>PNG</sword:formatNamespace>
<sword:formatNamespace>http://www.imsglobal.org/xsd/imsdp_v1p1
</sword:formatNamespace>
<sword:formatNamespace>PDF</sword:formatNamespace>
<sword:formatNamespace>METS</sword:formatNamespace>
<dcterms:abstract>This is the repository review. </dcterms:abstract>
</collection>

<collection href="http://eprints.ecs.soton.ac.uk/sword-app/deposit/inbox">
<atom:title>User Inbox</atom:title>
<accept>application/pdf</accept>
<accept>image/jpeg</accept>
<accept>text/xml</accept>
<accept>application/x-zip</accept>
<accept>image/png</accept>
```

```

<accept>image/x-png</accept><accept>application/xml</accept>
<accept>application/zip</accept>
<sword:collectionPolicy>This collection accepts packages from any registered users
on this repository.</sword:collectionPolicy>
<sword:treatment>Deposited items will remain in your user inbox until you manually
send them for reviewing.</sword:treatment>
<sword:mediation>true</sword:mediation>
<sword:formatNamespace>JPEG</sword:formatNamespace>
<sword:formatNamespace>http://www.loc.gov/METS/</sword:formatNamespace>
<sword:formatNamespace>IMS</sword:formatNamespace>
<sword:formatNamespace>http://eprints.org/ep2/data/2.0</sword:formatNamespace>
<sword:formatNamespace>PNG</sword:formatNamespace>
<sword:formatNamespace>http://www.msglobal.org/xsd/imsdp_v1p1
</sword:formatNamespace>
<sword:formatNamespace>PDF</sword:formatNamespace>
<sword:formatNamespace>METS</sword:formatNamespace>
<dcterms:abstract>This is your user inbox.</dcterms:abstract>
</collection></workspace><sword:level>1</sword:level>

<sword:verbose>>false</sword:verbose><sword:noOp>>false</sword:noOp></service>

```

A non-mediated (one that does not use X-On-Behalf-Of) authenticated deposit is then done in our case, given that the application runs on the user's machine and authentication can be provided personally. This can be done using an Asynchronous Javascript and XML (AJAX) script by either letting the application prompt the user for the username and password at each deposit

```
xmlhttp.open("POST", "http://sword.eprints.org/sword-app/deposit/inbox");
```

or by hard coding the *username* and *password*

```
xmlhttp.open("POST", "http://sword.eprints.org/sword-app/deposit/inbox", username, password);
```

The post is sent in an XML format with the content type specified and the file itself embedded encoded in base64. The standard fields such as filename (a number of files can be included in a single deposit specified as *< file >* entries under *< files >* as below), title, abstract and authors are also specified in this XML. For example the author, title and filename fields can be set in the XML as below.

```

.
.
.
*<creators_name>

```

```

    <item>
      <family>Mosweunyane</family>
      <given>G.</given>
    </item>
  </creators_name>*
<title>Promoting Public Data to the Semantic Web</title>*
.
.
.
<document>
<format>application/pdf</format>
<language>en</language>
<security>public</security>
<main>paper.pdf</main>
<files>
<file>
*<filename>paper.pdf</filename>*
<data encoding="base64"> JVBERi0xLjQNCiXk9tzfDQoxIDAgb2JqDQo8PCAvTGVuZ
RoIDIgMCBSDQogICAvRmlsdGVyIC9GbGF0ZUR1Y29kZQ0KPj4NCnN0cmVhbQ0KeJyVVctuG
zEMvBvwP/DcwObUayXAK0BHfA9gID+QB5BD...

```

The file is inserted programmatically using URLs, for example,

```
var fdata = new File("C:\\Documents and Settings\\User\\Desktop\\5 yr metadata study.pdf");
```

while the content type is set in the file post request to the server, for example,

```
fpr.setFile("POSTDATA", fdata, "application/pdf");
```

where *fpr* is the file post request object.

By implementing this protocol in the file browser on the desktop, the user is given the opportunity to publish data to the repository at the click of a button. The user can select files to be sent as they are browsing and collect them in the workspace panel as shown in figure 6.12. By using the “process collection” button below this panel the collected files with the respective metadata can be sent to the repository. The deposit can be completed later by the user with additional metadata that was not available with or could not be inferred from the files’ metadata.

Figure 6.13 shows the deposit in the user’s “inbox”, which is a set of data only visible to the depositing user (Allinson et al., 2008). Here further metadata can be added or the one added automatically edited (figure 6.14) after which the item can be posted to the “buffer” to await approval before being shown on the “live archive” in the EPrints repository.

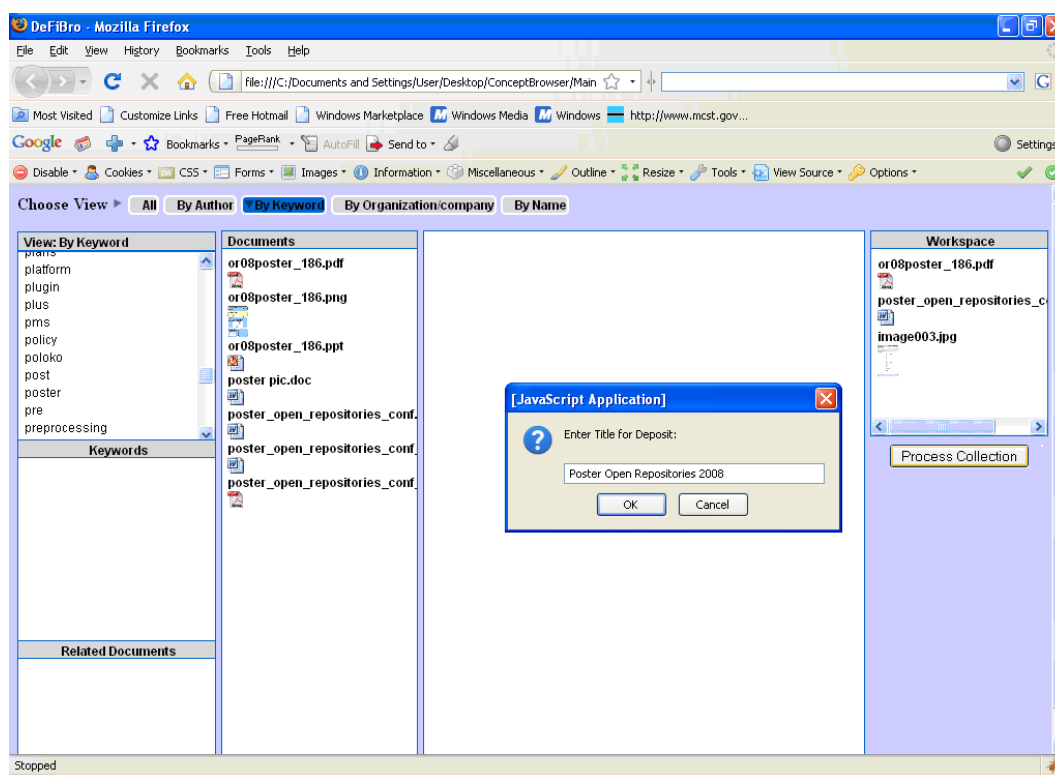


FIGURE 6.12: Entering the title to send collection of documents to EPrints

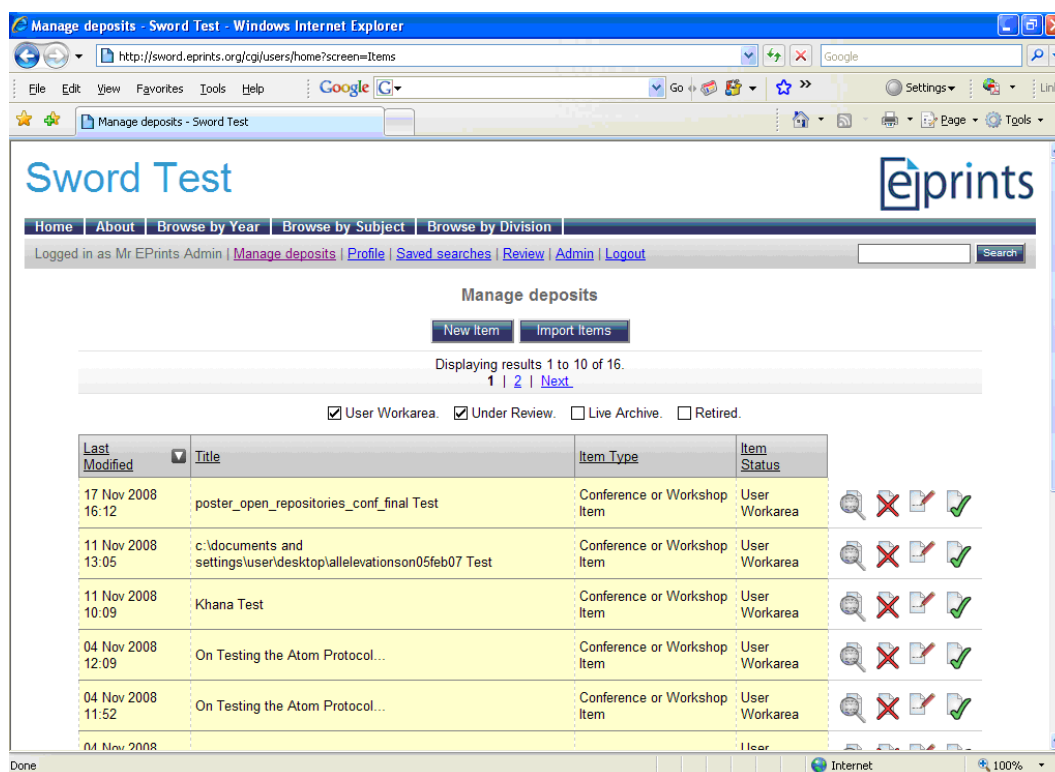


FIGURE 6.13: The Deposit in the EPrints Sword Test Repository

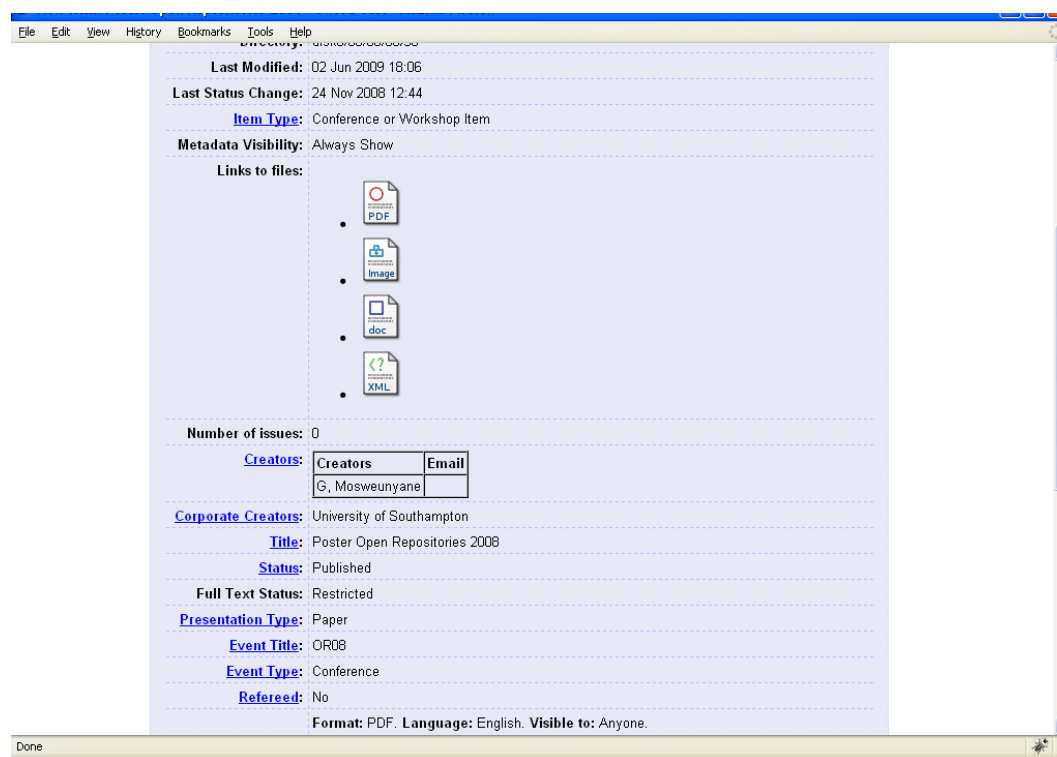


FIGURE 6.14: Metadata added with the document

6.5.3 Comparison to Other Projects

The described approach is similar in principle to the FeedForward project (Wilson and Potat, August 2009). The JISC project aimed to come up with a desktop tool that will be utilized by users to integrate content they created themselves with other collections in a simple workflow. In the FeedForward project publishing services are offered through a drag-and-drop interface (figure 6.15) and these include repositories and web 2.0 services. The project utilized the SWORD protocol for both these services and was used mainly to demonstrate the benefit of SWORD for repository interoperability. The platform was also used to demonstrate how personal information applications interaction with repositories can be treated in the same way as interaction with web 2.0 services.

The FeedForward project placed emphasis on integration of web tools, particularly web 2.0 ones and the desktop. This was done using a set of networked services to incorporate discovery, access use and publishing as part of the user's learning and research activities and interaction with the web.

6.6 Summary

This chapter has detailed out the steps taken in building a conceptual model for exploring desktop documents together with the instantiation of such a model through an

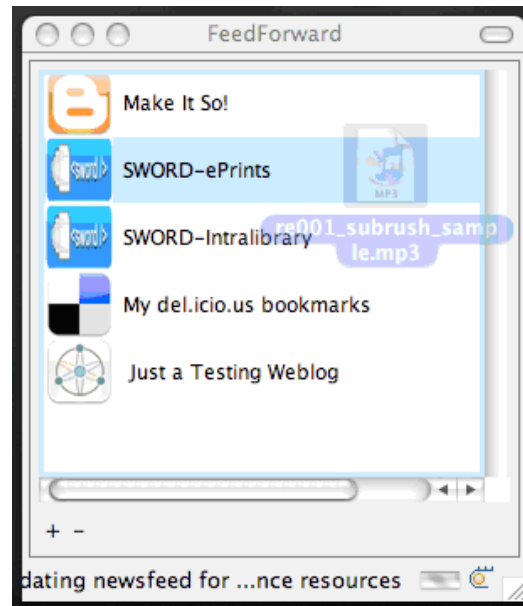


FIGURE 6.15: Dropping an item into a repository in the FeedFoward interface ([Wilson and Potat \(August 2009\)](#))

implemented interface. The components of the implementation and the interface have been discussed. The browser for desktop documents, DeFiBro, has been presented and use of its interface features explained. DeFiBro presents selected facets and the additional concepts to enable the user to dynamically organize their files. Organization can be according to the first letter of each file name, keywords, Author, Organization and details of a selected document are exposed through linked metadata, previews and related documents.

A model of exposing documents for retrieval and discovery based on indexing of their metadata terms has been presented. The model comprises of building a forward index for each document metadata based on its hierarchy information and other properties. All the indexes are then used to build an unified metadata index for the whole document set, which the user can browse. The index is also presented under a facet, and another alphabetic first letter of file name index is offered to browse.

The implementation utilizes terms that represent document identity and context and might be useful for recognising and retrieving documents. The interlinked nature of this information is presented to end users to utilize in browsing documents. The browsing structure provided is expected to integrate data (documents) in a different manner to what the users are used to, and provide an interface which will expose data which will have otherwise been “hidden” in a person’s data collection. The navigation structure provided through the interface is evaluated against system-provided navigation of the file system on the Microsoft Windows system using finding and associating tasks. This is presented in the next chapter.

Chapter 7

Evaluation - Hierarchy Navigation vs Metadata-based Navigation

7.1 Introduction

The implemented system endeavoured to demonstrate the importance of metadata and flattening of hierarchies in assisting users to locate, associate and discover documents while browsing their personal file hierarchies. To verify that users can indeed benefit from these, we perform an experiment that involves hands-on session tasks with users involving locating documents. To demonstrate the need to apply these concepts to improve operating system-assisted browsing of document hierarchies created by users on the desktop, we base these tasks on two environments. These are the Windows visualization method and our implementation, DeFiBro. With the Windows visualization method there are two ways: Windows Explorer or the simple zoomable visualization interface that involves clicking on folders to expand and view files and folders contained therein. Both have been found to have similar performance in locating tasks involving either familiar or unfamiliar hierarchies ([Golemati et al., 2007](#)). This is done in order to compare the two and to clarify the benefits of our approach through our implementation, while at the same time advocating for improvement of browsing documents on the desktop.

This chapter presents the process used to test and evaluate the implemented system. It begins with a description of the setup of the tasks used for the evaluation, how these tasks relate to the objectives of the research and explains the tasks together with the test users' details. The results of the test are then presented and analysed using statistical tests and retrieval systems' performance measures of precision and recall. Interpretation within

the context of personal documents' management then follows, followed by a discussion and conclusion.

7.2 Method of Evaluation

The research is mainly concerned with the browsing or navigation of personal information structures, in particular desktop document hierarchies, during knowledge tasks for retrieval of information sources (documents). [Kelly \(2006\)](#) recommends using naturalistic approaches that allow people to perform Personal Information Management (PIM) behaviors in familiar environments with familiar tools and collections. The document browser implemented will have to be therefore evaluated within the context of personal document hierarchies in settings that match as much as possible real user settings and tasks. The evaluation of Personal Information Management tools, however, presents some problems since they involve personal information, making it difficult to apply traditional evaluation approaches ([Gonçalves and Jorge, 2008](#)). Also tasks need to be specific to individual users using their own content such that enhancement of their memory and knowledge of their own data that the system is designed to support is studied ([Cutrell et al., 2006](#)).

A user-centred evaluation approach that involves tasks done by users will therefore be more appropriate, allowing us to evaluate the system's performance in real world settings with different users ([Dunlop, 2000](#)). This approach is carried out within some limitations discussed in the next section. A usability experiment to compare performance between the implementation and the users's usual way of navigating their hierarchy will help clarify the benefits of our approach. The experiment will measure the extent to which a set of test users can use the systems to achieve the goals in the specific context stated in the tasks. Efficiency, effectiveness and satisfaction are measures used for usability.

Effectiveness will measure the accuracy and completeness of the system which enables users to achieve the specified goals while efficiency extends this in terms of the amount of effort users put in, especially of resources such as effort, time, materials and cost. Satisfaction will be exhibited by the comfort and positive attributes users perceive from using the system. The interface also involves the use of index terms to retrieve relevant documents. Evaluation of retrieval systems is usually done using the precision and recall parameters ([Salton, 1989](#)). The index terms in DeFiBro are in essence queries that provide a way to retrieve information, and these parameters can therefore be applied in this instance.

Since the system was designed for the Windows environment it seems reasonable to compare it within the same environment with a similar way of browsing. Despite the fact that there are a lot of visualizations of file hierarchies on desktop systems, most users still use the Windows Explorer indented list paradigm (or its equivalent in Linux and Mac

operating systems) and the simple zoomable visualization that comes with Windows-based environments (Golemati et al., 2007) to browse their file hierarchies. It seems reasonable therefore to compare our system's performance with that of the Windows Visualization of the file hierarchy, namely either the simple zoomable visualization or Windows Explorer.

7.3 User Tasks - Context and Setup

7.3.1 Testing with a Test Hierarchy versus a User's Personal Hierarchy

Our implementation, DeFiBro, provides visualization for file hierarchies on the desktop, allowing a user to browse their documents. Similarly in Windows a user browses their hierarchical file structure through a visualization of folders and files. To test these systems we need a user's file hierarchy of folders and files to use. We could use either a selected test hierarchy or get the users to use their own hierarchies.

A user hierarchy is personal; only the user knows how it is organized. To come up with a task involving retrieval or locating of documents in that hierarchy will require an extensive inspection and study of each user's hierarchy to have knowledge of relevant documents that could be appropriate for the task, relations between them, as well as knowledge of their contents. Even then one cannot be sure of other parameters which can influence results, like whether the user knows or remembers the location of the file or not when given a task of locating a particular file. It would also be difficult, if not impossible, to devise a task which is similar across all users. Alternatively the researcher could ask the users to examine their documents and state which would be relevant. This is impractical and the users will not be willing to do it, and the study's results will be invalidated because the documents will have been brought to the user's attention (Gonçalves and Jorge, 2008).

Also for DeFiBro to be evaluated using the user's own personal data there is need for users to become familiarized and accustomed to the implementation by using it regularly over a reasonable period of time under observation and some well-thought out and thoroughly tested task given as a follow-up. This would give more insight into how it supports users' real tasks and goals within a work context over a long period of time (Dunlop, 2000). Given the limited time period for the research this could not be achieved in time to finish and write up the research, but will be discussed as a future work towards the end of this thesis.

Owing to the reasons mentioned above it was therefore decided that a test hierarchy will be used for the task. Using a test hierarchy ensures we have the same standardized settings across users and that the results are comparable. Although other researchers have observed that users have an intimacy with their personal data that cannot be

easily replicated with predetermined test sets ([Gonçalves and Jorge, 2008](#)), we still need to evaluate the system somehow within some confines that will give us an idea of how use in real life could be like. The hierarchy to be used therefore has to conform to several characteristics as listed below.

1. It has to be personal, created by a real user over a long period of time, so that it is as representative as possible of the group of situations the research is trying to provide a solution for. Though the results will be different from when using the user's own hierarchy as we cannot achieve a perfect set up, at least the test will be as close as possible to use in real life.
2. The hierarchy has to be familiar to the investigator or can be studied such that relevant documents are known in advance to facilitate setting up appropriate tasks.
3. It has to allow for free observation without fear of invading anybody's privacy.
4. It has to have documents at the three most levels in a hierarchy where users were found to store their data from the preliminary study, that is, levels 2, 3 and 4.

The conditions necessitated the investigator to use their own hierarchy for the tasks. Using the same task in the same context also has the advantage that the data will be relevant and consistent ([Garrido, 2008](#)). The hierarchy (see figure 7.1) had 19 top-level folders, 5579 files and 1526 documents (Microsoft Excel, Word, Powerpoint Presentations, pdf, htm and html). Whereas the mean number of files for users in the preliminary study was 39 102, which is much larger than 5579, this factor was disregarded in favour of a less sizeable but sensible file set for easier comprehension by users especially since file hierarchy was not their personal one.

7.3.2 Evaluation Tasks and Research Objectives

Before the user evaluation tasks could be devised it is necessary to revisit the research objectives, formulate research questions and hypothesis and how they could be tested through the user evaluation. The tasks main objective is to test the functional utility (usefulness) of DeFiBro, that is, if it can in fact be used for browsing and finding documents, and how well it satisfies the needs of users as stated in the hypothesis.

Our research had already established, through a preliminary user study, that most users encounter problems with effectively accessing and utilizing their personal file stores organized in the hierarchical folder structures supported by the current systems.

The objective of the research therefore was to present an alternative and better way of navigating these personal documents that were organized using the traditional hierarchical folder structures. The implementation showcases this approach which is evaluated through the statement of a hypothesis and a few research questions.

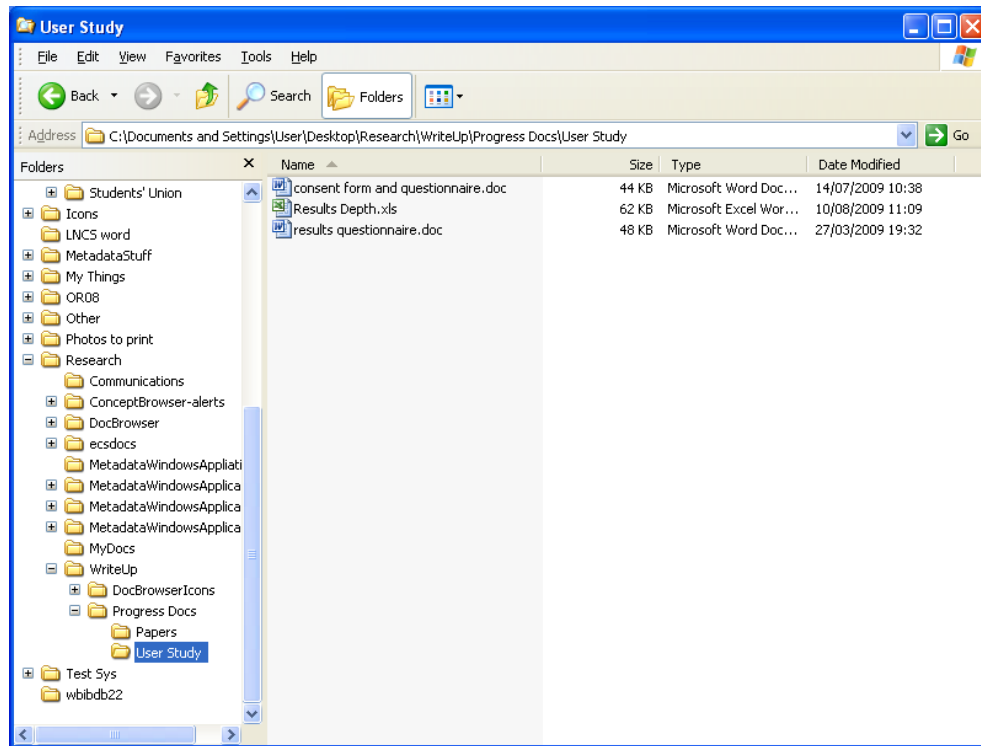


FIGURE 7.1: Part of the hierarchy on desktop showing folder documents used in task at level four

Our hypothesis is that navigation of desktop documents based on a flatter, linked metadata-derived browsing structure enables better access to documents such that they can be efficiently and effectively retrieved, associated and discovered.

The hypothesis can be restated in the form of research questions that can be answered individually through the evaluation.

1. *Finding* - Can users find the documents they are looking for using DeFiBro? How does this compare to the methods provided by the Microsoft Operating System?
2. *Associating* - Does the DeFiBro interface help in browsing and locating associated documents?
3. *Effective browsing, reminding and discovery* - Are users more likely to find and associate documents in their collection that they would otherwise be unable to do without the system?
4. *Utility of presentation of items on the interface* - Do users find the underlying organization (dynamic navigation) and other parts of the interface (linked metadata, previews, similar documents, collection basket) in DeFiBro useful?

The user tasks set and qualitative questions asked should therefore try to verify the hypothesis and get user feedback to answer these questions. The attributes measured

will include retrieval time, which will reflect the efficiency of the systems, as well as the percentages of located items, in case of tasks involving more than one item, to measure effectiveness of presenting and browsing associated items.

A number of tasks have to be formulated to dispel bias related to the location of a file, the file name or description, or to a test user having memorized the location of a file during the time given to study the hierarchy.

7.3.3 Test User Group

Test users for the evaluation were selected randomly from the same set used for the preliminary study. [Nielsen \(1993\)](#) and [Soken et al. \(1993\)](#) recommend using at least 5 test users, and preferably 8 or more for performance measurement evaluations in usability tests. However, even though our approach should include usability testing, it also involves comparative tests with personal information management tools. As such it is worthwhile to look at other appropriate recommendations and how similar studies have been conducted. Comparative tests involving evaluation of use of file hierarchies have used different numbers of users. For example, 15 users were used for comparing the use of the indented list and the zoomable user interface in Windows for file browsing by [Golemati et al. \(2007\)](#). [Turo and Johnson \(1992\)](#) used 12 participants to compare use of Treemap against the Unix tesh shell for directory browsing tasks. [Stasko et al. \(2000\)](#) on the other hand used 32 users for comparing Treemap against Sunburst for depicting file/directory hierarchies. [Kelly \(2006\)](#) recommends focusing on a small number of users in PIM case studies and using the findings to stimulate further studies.

Based on the recommendations and observations discussed above 10 participants were randomly selected out of the 25 that were involved in the preliminary study. These are people who have considerable computer skills and will therefore not need a lot of explanations or training regarding the use of Windows Visualization and DeFiBro, as already mentioned in the preliminary study (chapter 3 section 3.2).

7.3.4 Tasks Details

The tasks involved locating documents in the given test hierarchy using DeFiBro and Windows Visualization alternatively, given either the file name or a description of the document(s). The documents were located at the most popular levels where documents are stored as established by the user study of 2, 3 and 4. The last task which required location of multiple documents had some documents at level 1, in addition to level 2, to avoid frustrating users with having to “dig” deeper to find many files. The process was timed for each user and notes taken on the steps taken by the user to locate the document(s). The user was also encouraged to “think aloud” during the tasks so that

their thoughts and comments could be recorded to get further insight into their mental model of the system (Nielsen, 1993) and therefore their view of the system with regard to how it satisfies their needs.

The test users were asked to locate in turn a named file at level 2, two files in the same directory with different file names but same content given a description at level 2, named files at level 3 and 4 and finally a total of 7 files located in two different directories given a description that fitted all of them (most had the same content also) at levels 1 and 2. A summary of the tasks given is shown in table 7.1.

Task	Information given	Location in hierarchy	Number matches (files)	Other information
1(a)	Filename	Level 2	1	
1(b)	Description of document including some information about what it is about, location, and the time when it was needed	Level 2	2	Files located in the same directory
1(c)	Filename	Level 3	1	
1(d)	Filename	Level 4	1	
2	Description of criteria the documents must satisfy. Information given describes the content of the documents and time when it might have been used	Three at level 1 and four at level 2.	7	Files located in two different directories

TABLE 7.1: Summary of the Files and Hierarchy Levels Involved in User Tasks

Test users were given 5 minutes to study the test hierarchical folder structure (mainly top-level folders and subfolders to get an idea of what they contained) provided and given a brief overview and walk-through of the features provided in DeFiBro. They were also asked to alternate use of the two systems to average the bias associated with the results of one task in one environment influencing how the task is approached in the other system. However, the users were not instructed regarding which system to use first for each task nor was this recorded. It was only made sure that they used a different system after each locating task to minimise learning. The test environment was also reset at the beginning of each task to enable the timing to be started at the same point each time. The starting points were the desktop for Windows Visualization and the blank startup screen in DeFiBro.

The test users were given instructions on how to use Windows visualization and were only allowed to use the *search* facility in Windows only as a last option. In the cases where the search facility would be used, these times would be added to that of browsing to get the total time taken to locate the file.

At the end of the task users were asked to answer a few questions based on a five-point Likert scale and qualitative questions about how the two systems perform in comparison to each other. These questions were deliberately biased towards evaluating DeFiBro so that feedback could be solicited on how the implementation addresses their needs.

The user evaluation tasks and questionnaire are available in Appendix C.

7.4 Task Results and Analysis

7.4.1 Retrieval Times per Task

The results of the times the test users took to complete each task in the two environments were recorded in seconds and the averages computed. A depiction of the spread of the times taken to complete each task is shown in figures 7.2, 7.3, 7.4, 7.4, 7.5 and 7.6. The summary of the results is shown in figure 7.7.

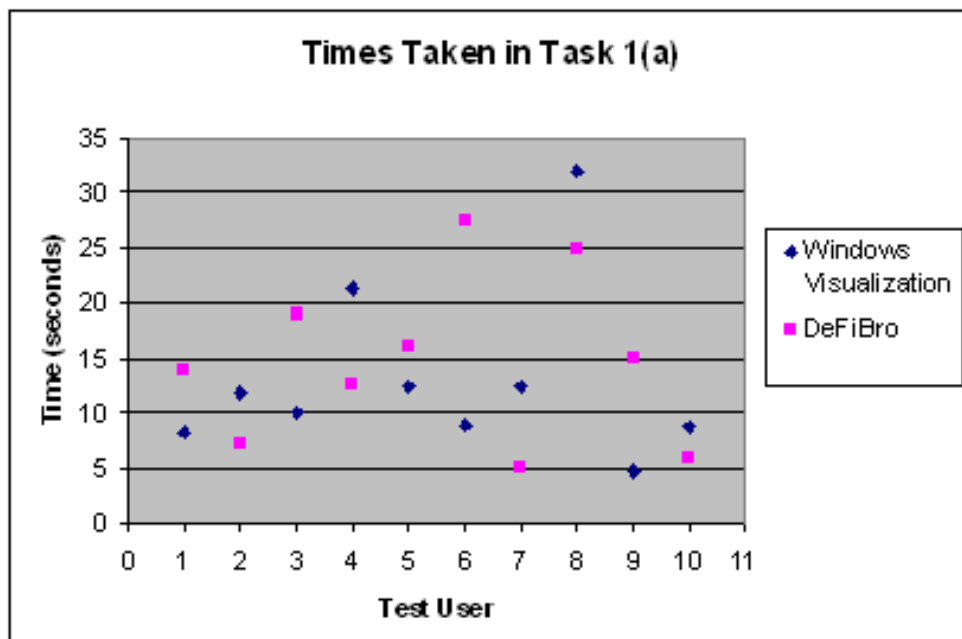


FIGURE 7.2: Locating a named document at level 2

A quick analysis of the results as depicted in these figure reveals that our implementation, DeFiBro, performed much better than Windows visualization in tasks 1(b), 1(c) and 1(d). This can be interpreted as meaning that users found our implementation more efficient to use in locating documents at levels 3 and 4, and to some extent level 2. Task 1(a) and 2 also involved locating files at level 2, so further analysis is needed to assess the effectiveness of the two environments from these measures. There is also need to verify and clarify the results' significance in relation to our objectives and hypothesis. This is carried out in the following section.

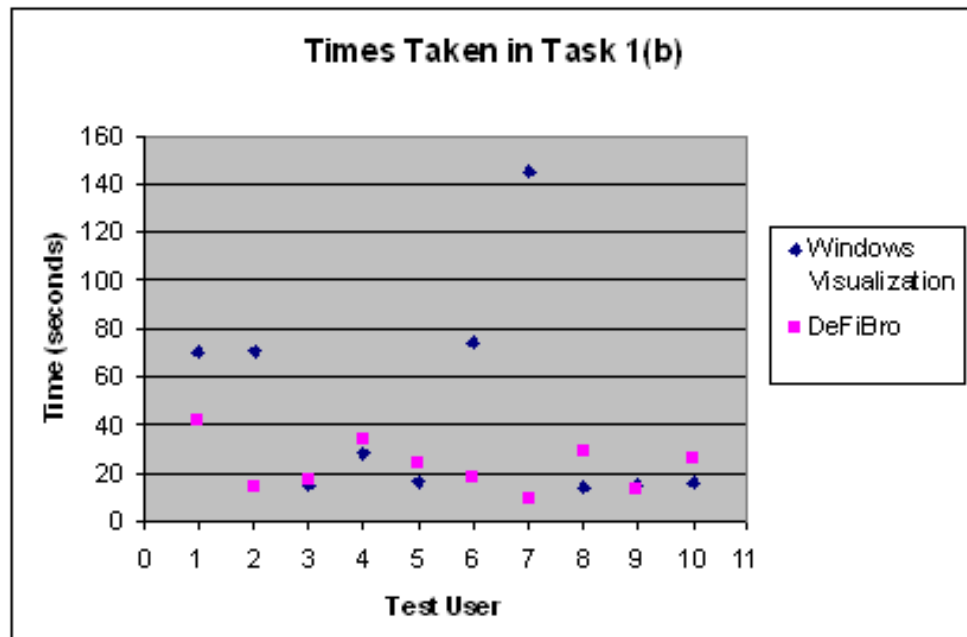


FIGURE 7.3: Locating two associated documents at level 2 given description

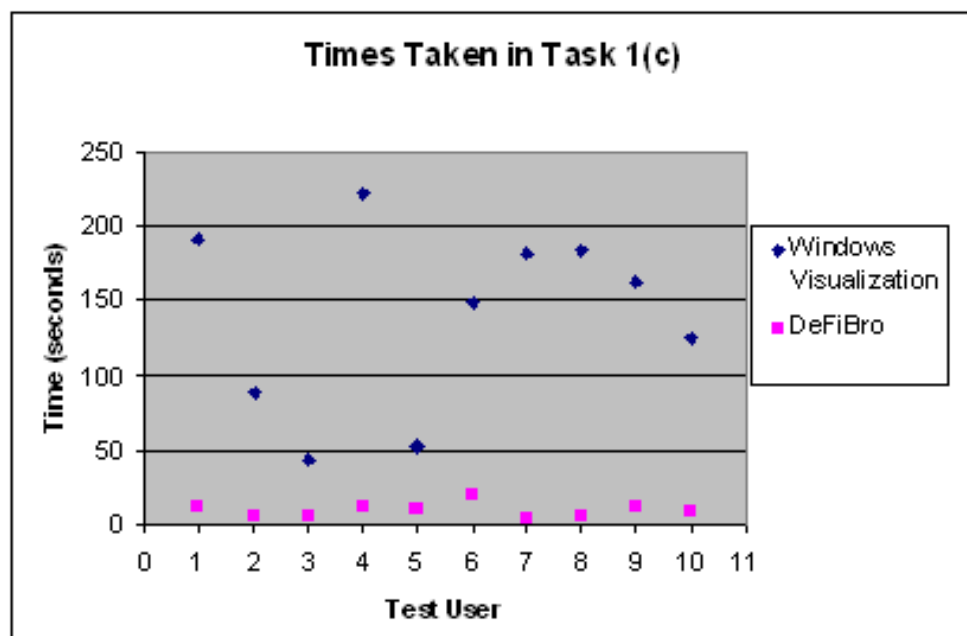


FIGURE 7.4: Locating a named document at level 3

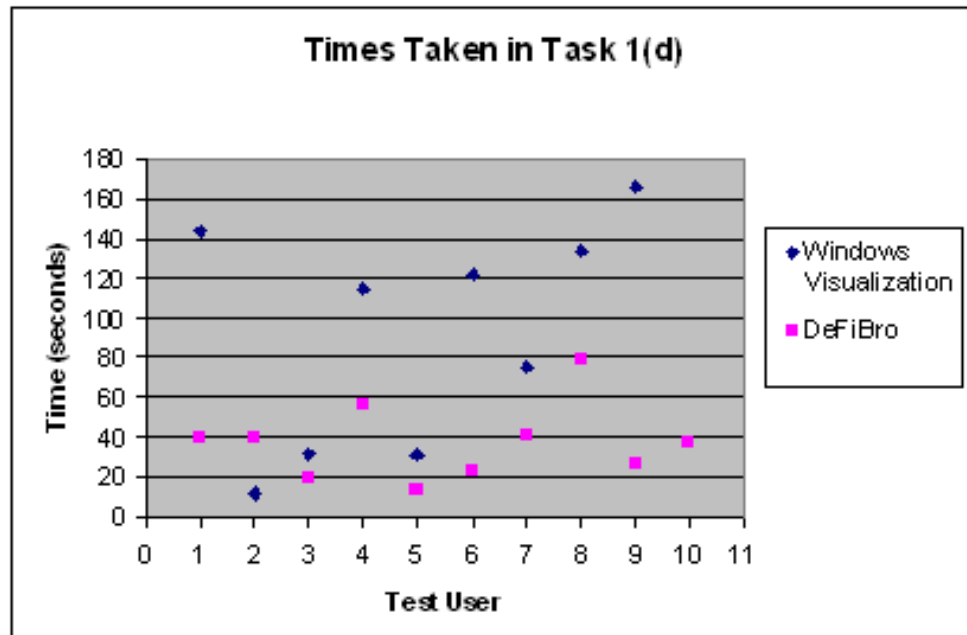


FIGURE 7.5: Locating a named document at level 4

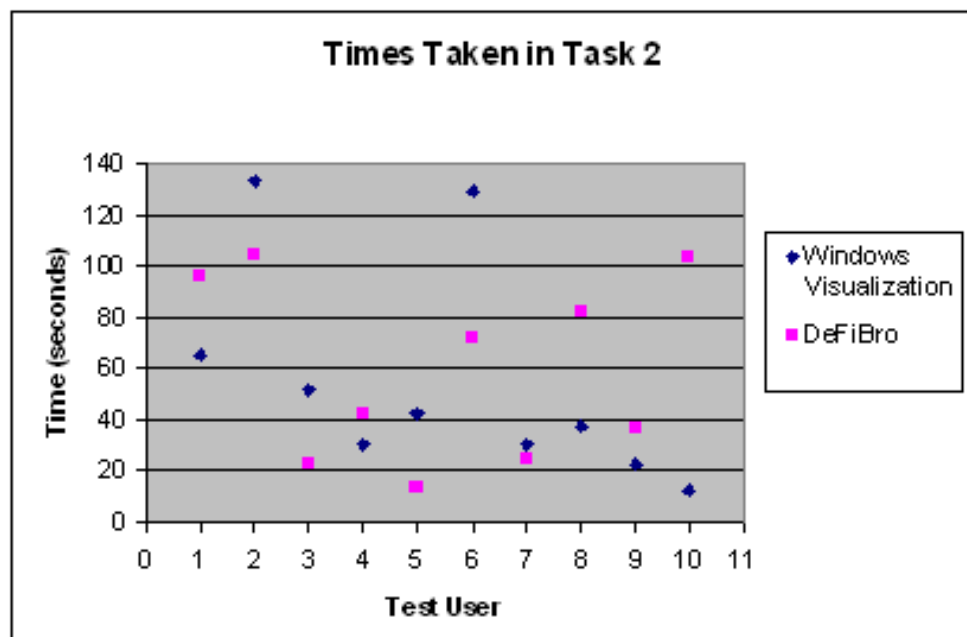


FIGURE 7.6: Locating seven associated documents in different directories given description

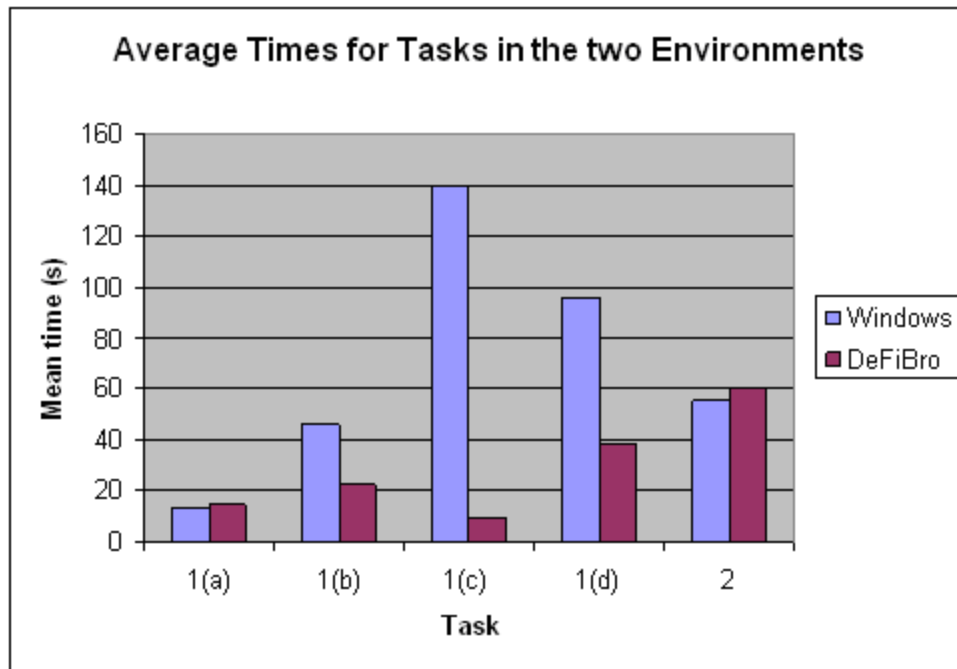


FIGURE 7.7: Comparisons of the Average Times for Windows Visualization and DeFiBro for all tasks

7.4.2 Hypothesis Testing by Statistical Significance Analysis of the Results

There is need to do tests on the results of the tasks to check whether the hypothesis is verified by these results. We perform Student's t -test ([Chatfield, 1983](#)), a statistical hypothesis test, to assess the likelihood of whether the results observed were just a chance occurrence or if there really is a significant improvement derived by using our implementation. We assume a normal distribution and use the t -test to test the null hypothesis. The **paired** version of the t -test is used in this case as the members of both data sets are matched, that is, the same participants were used for the same task in Windows visualization and in DeFiBro.

The t -test evaluation requires that we state several measures involved in the test.

- *The Research or alternate hypothesis.* Our research hypothesis states that our implementation (DeFiBro) offered users with better visualization and browsing, and therefore efficient and effective locating and discovery of desktop files compared to the Windows' operating system visualization of the same desktop files in hierarchies. The efficiency is based on measurements of time in tasks involving locating documents on the desktop.
- *The Null Hypothesis.* The null hypothesis is that there is no difference, in terms of benefits offered to users in visualization and browsing, between our implementation (DeFiBro) and Windows' visualization.

If the null hypothesis is true, the test statistic t should follow a t -distribution with $n - 1$ degrees of freedom.

- *The Probability of error level (alpha level).* Because of our small sample size (10 participants), we will assume an error level of 5% that there is a relationship between the two systems being compared when there really is no relationship. The p -value gives the probability of the result assuming the null hypothesis. The results of each p -value for each task will therefore be assessed against the 0.05 level. If the p -value associated with t is low (less than 0.05), then the null hypothesis will be rejected in favor of the research hypothesis.
- *The Degree of Freedom.* The sample size n for all the tasks is 10 so we calculate the Degree of Freedom (df) as $df = (n - 1) = 9$.

The test statistic t computed as

$$\frac{m_1 - m_2}{\sqrt{\frac{s_1^2}{df} + \frac{s_2^2}{df}}} \quad (7.1)$$

Where

$$m_i$$

is the mean,

$$S_i$$

is the standard deviation and df is the degree of freedom.

The t -test is then performed using these measures. Once a t value is determined, a p -value can be found using a table of values from Student's t -distribution, which is available in the appendix of most statistical textbooks ([Trochim, 2006](#)). However, one can also use statistical computer programs widely available to calculate the p -value.

We use Microsoft Excel's TTEST function ([Microsoft Corporation, 2009a](#)) to calculate the p -value for each task. The t -test will be a one-tailed test for significance because our null hypothesis is that the mean times will be higher when using DeFiBro than when using Windows visualization. The Excel TTEST function uses the formula

$$TTEST(array1, array2, tails, type) \quad (7.2)$$

where *array1* is the first data set, *array2* is the second data set, *tails* specifies the number of distribution tails (1 for a one-tailed distribution and 2 for a two-tailed distribution) and *type* is the kind of t -test to perform (1 for paired, 2 for two sample equal variance and 3 for two sample unequal variance).

Table 7.2 shows the variables involved for the paired evaluations for each task. The mean, standard deviation and variance for each set are also calculated and shown in the table.

Task	Tool	Mean (s)	Standard Deviation	Variance	t	p -value
1(a)	Windows Vis.	13.03	7.52	56.58	-0.58	0.28
	DeFiBro	14.73	7.21	52.01		
1(b)	Windows Vis.	46.37	41.12	1690.51	1.57	0.074
	DeFiBro	22.62	9.73	94.58		
1(c)	Windows Vis.	139.93	57.90	3352.20	6.88	0.000036
	DeFiBro	9.44	4.50	20.22		
1(d)	Windows Vis.	96.02	51.29	2630.97	3.63	0.0027
	DeFiBro	37.97	18.06	326.31		
2	Windows Vis.	55.48	40.24	1619.28	-0.31	0.38
	DeFiBro	59.74	34	1156.00		

TABLE 7.2: Summary of Calculated Times and t -test Results for all Tasks

From the results we can assess whether to accept or reject the null hypothesis for each task by looking at either the test statistic t or the p -value. From the Percentage points of Student's t -distribution table (Chatfield, 1983), for a one-tailed test (DeFiBro better than Windows visualization), an error level of 0.05 and 9 degrees of freedom $t_{0.05,9} = 1.83$

Thus $P(|t| > 1.83) = 0.05$.

If therefore the t calculated for each task is greater than 1.83 then there is reasonable evidence in the result to assume that the null hypothesis is not true.

On the other hand we can use the p -value to assess the likelihood of the result assuming the null hypothesis. Like it is stated above if the p -value associated with t is lower than our assumed level of 0.05, then there is reasonable evidence to reject the null hypothesis and assume our alternate hypothesis.

The results from table 7.2 clearly show that the null hypothesis should be kept for tasks 1(a), 1(b) and 2 and should be rejected for tasks 1(c) and 1(d). We therefore assume our research hypothesis for tasks 1(c) and (d).

7.4.3 Precision and Recall for Multi-Document Task Results

Tasks 1(b) and 2 involved locating one or more files and though all the users completed the tasks, some did not do so successfully in that they only identified a fraction of the documents satisfying the criteria (for example, some found only 2 out of 7 documents). This can better be explained by the retrieval system concepts of *precision* and *recall*. Because the systems being investigated involve browsing, which is essentially a retrieval task, these concepts may be applied to measure and explain the systems effectiveness. However, the use of precision and recall in this context is not exactly the same as they are used in information retrieval, that is, the definitions had to be adopted to fit our purpose.

Precision in retrieval systems is defined as the proportion of retrieved documents that are relevant while **recall** is the proportion of relevant documents that are retrieved by the system (Singhal, 2001). Whereas these concepts can be easily applied in DeFiBro their meaning cannot be applied directly or interpreted easily in the context of browsing hierarchies. In DeFiBro a user selects an attribute value to browse by and the system presents a visualization of files from which they can select the relevant one. The method the user uses to find a document could therefore be noted easily during the task and the process repeated easily afterwards to derive the parameters for calculating precision and recall. In Windows visualization of the hierarchical file structure, however, the process involves navigating folders; going through and inspecting files and/or subfolders and sometimes even backtracking/traversing backwards until the file is found. Such a scenario is not easily comparable to that in DeFiBro because the navigation process involves folders/subfolders which are not presented in DeFiBro but whose presence may impact on the retrieval process but are not included in the definitions of precision and recall. Some users also gave up using Windows visualisation when they could not find the documents requested and resorted to using Windows search. This, and the traversal through folders and encountering of documents along the way makes it difficult to explain precision in Windows visualization.

Given N as the set of documents that are relevant and $i(t)$ as the set of documents retrieved by use of index term t

$$Precision(t) = \frac{|N \cap i(t)|}{|i(t)|} \quad (7.3)$$

$$Recall(t) = \frac{|N \cap i(t)|}{|N|} \quad (7.4)$$

For task 1(b) the user was given information about files to find at level 2. There were two files in the same folder with the same content that matched the criteria given. Precision in DeFiBro based on the browsing paths followed by the users for the task varied from

0.02 to 0.67 (average 0.50), depending on which attribute one used to find the document (table 7.3). The average recall was the same for both systems, which disputes our hypothesis that previews will help in discovering documents. It was however noted that the previews did help in quickly checking out whether a chosen document the correct one, hence the lower mean time for DeFiBro as seen in table 7.4.

User	DeFiBro		Windows Vis.
	Precision	Recall	Recall
1	0.67	1	1
2	0.33	1	1
3	0.67	1	0.5
4	0.167	0.5	0.5
5	0.33	0.5	0.5
6	0.167	0.5	0.5
7	0.67	0.5	0.5
8	0.67	0.5	1
9	0.02	0.5	0.5
10	0.67	0.5	0.5
Average	0.50	0.65	0.65

TABLE 7.3: Precision and recall results for task 1(b)

For task 2, which involved retrieval of a group of related files, the average precision in DeFiBro was 0.58. Although the time results in table 7.6 show no significant difference between the two systems, average recall results show DeFiBro performed much better at 0.76 recall compared to Windows Visualization's 0.43. The results are shown in table 7.5. This could be attributed to the precision in DeFiBro being higher (presentation of documents in one interface rather than traversing folders in Windows visualisation), implying that the browsing terms provided offered better clustering and association of files compared to the "scattering" of files in the hierarchical file structure.

7.4.4 Users' Approach to the Tasks and Feedback

7.4.4.1 Browsing Behaviour in Windows Visualization

During the task some observations were made regarding how the users approached the task of locating a file in the hierarchical folder structure. Whereas some users browsed

User	Win Vis (sec)	DeFiBro (sec)
1	69.67	41.74
2	71.12	13.84
3	14.66	17.59
4	27.94	34.25
5	16.83	24.07
6	73.95	18.01
7	145.23	9.18
8	14.05	28.89
9	14.79	13.08
10	15.45	25.51
Mean	46.37	22.62

TABLE 7.4: Time results for task 1(b)

fairly quickly and without much effort to check through and locate some documents, some took their time to check and verify the documents by either opening or checking the thumbnail preview if available. The problem with those who browsed quickly though was that they usually failed to do thorough checks to make sure they recovered all the documents required for a task.

An observation was made that most of the users had strong opinions of where the document they were asked to retrieve should be located in the hierarchy and this had an impact on the way they browsed to find the document. They expressed this by verbalising their thoughts (“the file must be in the *research* folder!”) and concentrating on the said location instead of browsing other parts of the hierarchy, especially for deeply located documents. Almost all users doing this were right about the top-level folder, but spent a lot of time browsing and concentrating on the first two levels of the hierarchy even if the file was not there. They did not make a good effort to browse through and check the sub folders, especially if there were files in addition to sub folders at that level. This could be interpreted to mean that users are subconsciously comfortable locating files in the first and second level of the hierarchy, and that although they do store files at deeper levels (as established in the preliminary study) they need help locating those items located deep in the hierarchy. Concentrating on the folders than on the files by users also seems to corroborate the results of an evaluation done on Windows

User	DeFiBro		Windows Vis.
	Precision	Recall	Recall
1	0.24	1	0.43
2	0.13	1	1
3	1	1	0.43
4	1	1	0.43
5	1	1	0.43
6	0.04	0.43	0.14
7	0.2	0.43	0.43
8	0.2	0.43	0.14
9	1	0.43	0.43
10	1	0.86	0.43
Average	0.58	0.76	0.43

TABLE 7.5: Precision and recall results for task 2

Visualization by [Golemati et al. \(2007\)](#) that found that users were better at locating folders than files and concluded that it is probably because users pay more attention to the naming of folders as conceptual categories in their organization than files.

7.4.4.2 Browsing Behaviour in DeFiBro

- Use of Filename Index - The alphabetic index was utilized by almost all users when the filename was known. However, a couple of users tried to use it to find filenames including a word with a letter starting with a chosen index (for example filename containing “april”). This may indicate a need for further expansion (or combination of this with term index) of the index based on analysis and inclusion of terms already derived and from numeric dates.
- Use of Metadata Index - While using DeFiBro a couple of users preferred to stick to one method of finding documents. Both used the filename each time, even for tasks for which the filename was not given. They tried to guess the filename in such cases. This pattern was followed after the method was found to be successful in the first couple of tasks, or that the users were either comfortable with it or it was similar to the method they use to find their own documents.

User	Win Vis (sec)	DeFiBro (sec)
1	65.08	95.39
2	132.9	104.4
3	51.49	22.71
4	30.87	42.49
5	42.38	13.39
6	129.12	71.82
7	30.44	24.82
8	37.59	82.19
9	22.55	36.86
10	12.36	103.36
Mean	55.48	59.74

TABLE 7.6: Time results for task 2

Most test users verbalized very simple property value (for example “poster”, “conference”, “visa”) about the file they were asked to locate and could easily relate these to and follow the index provided as keywords in the interface to locate the file. However, getting to the end results for a task for most users was delayed by inspecting other things on the screen that had to do with the documents while browsing, for example previews and keywords. In other words the users were distracted by the fact that other things were provided rather than concentrating on finding the documents straightaway. As much as this was bad for efficiency during the task it could also be useful for effectiveness as exemplified by the better recall ratio discussed earlier.

Users also used previews to check if the document is the right one.

Users also tried several terms in the index (the keywords list) when trying to find the files while exploring, and after a few attempts were usually able to find them, compared to Win Vis where they took a long time searching through, giving up in some instances and ending up using Windows search.

- Use of Author and Organization Facets - Only a couple of users tried in single instances to locate documents through the other facets. One such attempt during the task involving finding the Korean visa application form was successful (using “Korean Consulate” as organization). The other attempt involved browsing

through the first few authors and giving up after failing to locate the document in the document sets presented.

Though interest in the use of these facets was low, they are predicted to be more useful in the user's own hierarchy as the familiarity with one own data can make a big difference. Users can associate more with the authors and organizations in their collections and these could serve as memory aids in retrieval.

7.4.4.3 Questionnaire Results

As mentioned in section 7.3.4 a short questionnaire was given at the end of the user evaluation to solicit views about the utility, usefulness and usability of the system. The results of the quantitative part of the questionnaire are shown in figure 7.8.

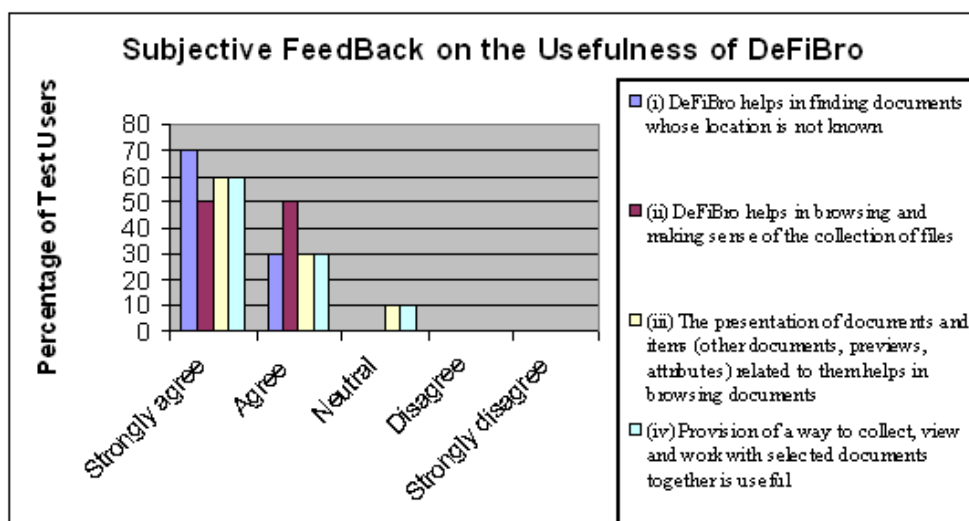


FIGURE 7.8: User feedback on the Usefulness of DeFiBro

The statements measured the users' emotion towards the use of the implementation regarding help with locating documents, browsing collections of documents and the presentation and functionality provided.

All the users either agreed or strongly agreed that DeFiBro helps them in browsing file sets and in finding documents whose location is not known. 90% of the users agreed or strongly agreed that other functionality (related documents, previews, attribute values and a "basket" for collecting documents to work with) provided in addition to a list of documents was useful in helping them browse and work with their documents. Only one user was not sure whether these will be useful or not. None of the test users disagreed that the system was useful.

Two questions were provided on the qualitative part of the questionnaire to gather views on the positive and missing aspects of the system as viewed by the test users. The answers to these questions provide further insight into the mental model of users during

the browsing tasks and the utility of the implementation in relation to the functionality they are used to or would like to be provided with in working with their documents.

Users reported the positive aspects of DeFiBro as being user friendly, offering of browsing by association and “searching” without providing search terms as well as helping out in a “badly organized” situation and providing features that enable more effective browsing like previews.

Suggestions for improvement by test users were dominated by the need to quickly move within the results (documents found by browsing, keywords, organizations, author) by automatic selection/shortcuts based on keyboard characters. Other suggestions were for the cosmetic appearance of interface to be improved through conversion of the interface to a more graphic one, showing attribute-value pairs instead of only attributes and providing other attributes to locate files such as date.

About half of the users indicated that they would have like the system to somehow be connected to the file hierarchy to provide for a way to get to the original location of the file. One user commented “...because in the first place I was trying to locate the file” to emphasize the need for such a system to be intertwined with the original way the files were organized and possibly the sense of attachment the users have with the way their “organized” file structure.

7.5 Discussion

7.5.1 Improving Efficiency of Browsing and Locating Documents

The efficiency of the two environments was measured by its performance in relation to the users’ speed and effort expended in locating the documents in the tasks given. Time taken to locate documents in the given tasks was used as a measure of these attributes.

The results show that DeFiBro performed reasonably better than Windows Visualization for tasks involving locating files at deeper levels (in this case levels 3 and 4 in the hierarchy which was demonstrated in tasks 1 (c) and (d) respectively) than at shallow levels (level 2 as demonstrated in tasks 1(a), 1(b) and 2, and level 1 demonstrated in task 2). This can be attributed to the fact that in DeFiBro the hierarchy has been flattened to allow the user to reach a file directly by using different attributes users easily associate with a given document keywords and other properties related to the file rather than the “search-by-location” offered by the Windows Visualization. To reach deep-level folders requires time and from observation of users performing the tasks, is more difficult by just traversing the hierarchy than by browsing some attribute values derived from the file properties.

The two systems' performance is more or less the same when the files are located in fewer folders. This is because of the advantage of reaching the file by browsing fewer folders, and therefore taking less time, in Windows visualization that is comparable to deciding an attribute to browse, selecting it and checking among the files presented in DeFiBro. Nevertheless DeFiBro was seen to be still better than Windows' Visualization in other ways and this is discussed in next section.

7.5.2 Improving Effectiveness of Browsing

Effectiveness in the two environments is reflected by the usefulness and helpfulness while carrying out the tasks. Two of tasks using the lower levels of the hierarchy were also set up for users to find *associated* documents. The usefulness and helpfulness of the two environments could therefore be measured by the way they *exposed*, *presented* and *reminded* users about the "group" the document belonged to.

The precision and recall measure was used to further evaluate these tasks for which the null hypothesis was proven by statistical analysis, that is, where it was found that the two systems performed more or less the same in terms of the times taken to locate documents. In these cases we could have concluded that there was no difference in terms of the benefit of browsing and visualization in the two environments. This would have presented an incomplete picture of the evaluation as we could have only tested *efficiency*, and not *effectiveness* of the two in comparison to each other. Precision and recall allowed us to test effectiveness as stated above.

The results showed that in one task for a higher precision in Windows Visualization, DeFiBro had at least the same recall while in another both precision and recall were much higher in DeFiBro (almost twice as much). This leads us to conclude that at least for the given tasks the visualization in DeFiBro is more effective in presenting to and reminding users' of documents in the hierarchy to ease locating and discovery of documents than the navigation of hierarchical file structure itself.

7.5.3 Other Issues

The task was based on unfamiliar hierarchy that was not the user's own. Even though the users were given time to study the hierarchy, it was only for a few minutes and they did not get to study the files and all the sub folders, which minimized learning of the hierarchy. We can relate these results to cases where the user have not seen their documents for some time, have forgotten their hierarchical structure, use of ambiguous folder names in the hierarchy, or with users dealing with a hierarchy that they did not create themselves or one that they acquired from somewhere else. In such cases, as the results show, users will have a hard time locating deeper-level files than ones stored in fewer sub folders.

The other factor involved is the scalability of the system in relation to the data set. Like it was discussed in section 7.3.1 the number of files used in the test was only a fraction of the average number of files the users in the preliminary study had. Although this was a typical user's hierarchy it might not have been representative of situations where users have a larger amount of files in terms of results presented by DeFiBro. The tasks results with particular users, however, revealed a couple of situations where a lot of documents were presented in DeFiBro when using some browsing terms, indicating that situations might arise where many documents are presented as results. In such cases there might be need to provide means to refine the search by using other criteria to search within the results, which is not provided in the current system.

The user's skill level is also very important and could contribute to how they deal with the task, and in particular, their own file hierarchies. However the setup of the task decreases the level of bias in the task given by requiring the same task to be undertaken in both the Windows settings and the DeFiBro interface by the same user and averaging the results across users.

The approach adopted for the evaluation has mainly been concerned with whether and how the system meets the user's needs, which is the approach usually adopted in information seeking evaluations (Kules et al., 2008). Although time was used as the basis for evaluation, it may not accurately reflect the actual benefit of the exploratory interface, as it has been discovered before that a longer time might indicate more beneficial browsing in terms of discovering relevant material (Capra et al., 2007). An extended user evaluation with the user's hierarchy and follow-up over a longer time might then be more appropriate.

In summary the tasks results show that navigation of desktop document hierarchies can be considerably aided by exploiting the knowledge embedded in document properties. The knowledge is captured in multiple attributes and when "mined" and presented in a comprehensive framework to users can help

1. *Allow browsing to take memory load off user by presenting reminders* - Present documents in a way which exposes them better for easy location by offering several properties that could be used to locate them, their previews and their properties rather than requiring users to remember locations in hierarchy or search terms to use.
2. *Associations* - Users more likely to come across more relevant documents when they browse using attributes rather than when they simply traverse hierarchy. Remind users of documents that could be considered to be in the same group by presenting them together using attributes related to them.
3. *Discovery* - Forgotten and deeply-located documents more likely to be found through knowledge-based browsing than traversing hierarchy.

7.6 Summary

In this chapter a method used to evaluate the implementation was presented and justified. The tasks used for the evaluation were devised to test efficiency and effectiveness of the implementation in comparison to the regular way of browsing hierarchies. A test hierarchy was chosen in favour of users' personal hierarchies for purposes of consistency between users to allow the results to be assessed as a group and to be compared. The tasks involved retrieval (locating) of documents at different levels in the hierarchy. Subjective feedback was sourced by means of a questionnaire at the end of the evaluation tasks.

Although it is a small scale test, the results of the tasks indicate an advantage in using DeFiBro in terms of efficiency and effectiveness of browsing compared to Windows visualization method for browsing hierarchies. Users also reported a richer browsing experience offered by multiple, dynamic ways to recover documents and view related items in DeFiBro. This indicates that users can thus benefit from the provision of such structures to improve their efficiency and effectiveness which will in turn result in higher productivity in their everyday tasks. To come up with a trusted and sufficient interface that could be recommended for use would however require more research, testing and evaluation with users. One possible scenario could be to enhance the existing operating system navigation structures with the main features of DeFiBro; indexes, previews, relations and extraction. This and other possible directions are discussed after the summary in the next chapter.

Chapter 8

Concluding Remarks and Further Work

8.1 Summary

This thesis has emphasized further the idea already presented by researchers that computers possess enough processing power to offer better support to users and minimize user-memory load in information-seeking and integration tasks. It mainly focuses on tasks involving retrieval and discovery of documents to support knowledge creation and integration on the desktop.

The main novel contribution of this thesis is a recommendation on how to augment navigation of personal file (and possibly other labelled) hierarchies, which will in turn augment user memory in knowledge-seeking and building tasks involving documents on the desktop. The recommendation is based on a validated solution encompassing metadata extraction, semantic integration, indexing and clustering to provide flexible faceted, linked, similarity- and preview-based browsing. A model for desktop documents browsing was first developed as a backbone for access after further insight into filtering and browsing documents was provided by experiments done with semantic metadata. The model makes use of terms extracted from the metadata values and similarity calculation between documents based on these terms. The model was then instantiated on the implemented interface (DeFiBro).

The design of DeFiBro's interface and development of the model were informed by a preliminary study and a literature review. The aim of the preliminary study was to verify the hypothesis that users encounter limitations with desktop documents' access with system-provided tools, and the number and extent of the spread of documents in the users' file hierarchies. The literature review explored existing solutions and design of information-seeking solutions, which helped inform the design of the solution.

The exposure of metadata-derived terms in DeFiBro was hypothesized to reduce the user's cognitive load during access and enable efficient and effective retrieval and discovery of desktop documents. A comparative evaluation of the effectiveness and efficiency of the new interface against system-provided file hierarchy navigation based on time taken to complete set tasks was therefore done to verify this hypothesis. The results suggest a statistically significant ($p=0.5$) advantage towards DeFiBro in tasks involving locating documents stored deep in the hierarchy. Effectiveness and discovery was measured through the extent of completion of the tasks (how many documents were found against the relevant ones) due to presentation in both interfaces. DeFiBro was found to be better than Windows Visualization in this respect. The advantage in DeFiBro is attributed to the exposure of linked index terms and associations between the documents.

Other contributions are a publishing method of desktop documents with metadata to other organized stores like the institutional repository and further insight into user's strategies in storing files on the desktop. The publishing method is included in DeFiBro as a feature of the interface that enables selection and extraction of documents during exploration. Methods of distributing documents amongst folders in the hierarchy are described as a results of the user study conducted.

Information accessibility on the desktop was explored at the initial stages, specifically information stored in documents in files on the hierarchical folder structures. The view taken is that users have created knowledge structures on the desktop in the form of these file hierarchies. Operating system-provided access methods have been seen to provide limited support for finding, discovery and association-navigation of documents across this structure. Solutions like adoption of a different metaphor for the desktop and change to organization methods provided to users have been proposed to counteract these defects but have not taken off on a general scale or been adopted by common operating systems. This means that until operating systems provide different support for document management, support should be focused on users' organization in hierarchies and reward for their effort in terms of enhancement to retrieval and discovery methods based on the user-provided attributes which are more likely to match the user's mental model and therefore be more effective. With processing and exposure of rich metadata already available as document properties the navigation of these structures can be improved to make it easier for users to browse, retrieve and discover documents 'hidden' in their hierarchies.

The importance of document metadata in exposing the desktop documents for browsing to supplement the use of system-supported navigation is especially emphasized in this research. Regular document attributes form the centre of the solution offered. The browsing structure presented makes use of rich metadata that is already available and accessible in the form of file properties. In most operating systems these are packaged with the documents and are available for viewing, editing and extraction as the *built-in*

file properties, and are commonly indexed together with documents' contents and utilized in search systems. The information embedded in this file metadata was extracted, structured as semantic data and exploited to derive concepts and enhance facets to offer a comprehensive framework for exploring documents. Structuring metadata using Semantic Web technology enables easier integration, extensibility and access of the data from different applications.

The approach taken to investigate and validate these assumptions was presented in this report. First a survey of operating system-supported organization and access methods for files on the desktop was undertaken as a basis for problem-solving. A literature review of other approaches taken to improve such organization and access followed, including recent proposed integration and data structuring methods. To validate the view taken regarding the shortfalls of operating system methods and to build a view of the extent of the problem, a study of several personal file hierarchies was undertaken. This, together with the experiments undertaken and a review of information-seeking behaviour and recommended solutions, helped inform the approach taken in the final solution.

8.2 Further Work

8.2.1 Metadata Extraction and Presentation

Metadata in the form of file properties forms a basis for this research. Currently this metadata is not stored in a standardized manner across different file formats, making it difficult for sharing and processing by applications and document stores. For example, for this project different documents that were encountered (Microsoft Office, Portable Document Format, JPEG, html) had different sets of metadata that had to be sourced differently. Structuring and storing the metadata in Semantic Web form by operating systems might help counteract this. Microsoft, for example, has recently come up with tools aimed at capturing metadata in Word documents at authoring time ¹¹ for transfer with the document to recognised repositories. File metadata is also already being indexed by systems such as Spotlight in Macintosh systems. If the same procedure is followed by all applications for capturing and storage of this metadata in Semantic Web form the system implemented could be easily fed with the different sets and evaluated. However if this is not possible, the same approach taken to source file properties and an investigation into how this could be done across applications can be followed. This extraction and evaluation may help further analyse its usefulness for retrieval and discovery of documents, and possibly inform further enhancements to the solution.

¹¹<http://research.microsoft.com/en-us/projects/authoring/> [last accessed 28/09/09]

8.2.2 Use of Extended Attributes

In addition to the built-in properties provided by the operating systems, the user can define and add their own properties (called *custom* properties) to a document. These name/value pairs can be added to files, and also to directories in some operating systems. This project has so far only dealt with only the built-in document properties. For productivity applications such as Microsoft Word this functionality was provided mainly such that users can specify recurring field values only once such that they could be inserted it into the document at desired locations with modifications done at one place. These may however use these to select files associated with a given attribute.

If all filesystems and applications support extended attributes then they can be used to specify groupings between the files and directories and shown together with the hierarchical file structure and in applications.

8.2.3 Browsing Google Results

The importance of browsing individual terms to locate desired results is even shown in Web browsers such as Google ¹². The browser requires the specification of a search term or phrase to come up with results and in addition the google toolbar has a feature which allows the user to explore the results based on the individual terms in the search phrase (figure 8.1). Using the *highlight* and *Word find* buttons one can highlight all the matching results or find instances of given words in the results respectively.

Though this feature might be useful primarily for navigation inside a web page, it could be useful if it follows the same method used for discovery of documents in this research. The most relevant documents' metadata (as opposed to content use in addition to metadata) could be extracted and the index terms exposed to provide the user with a different view of results. The metadata that could be used in this regard are the HTML title element or the image alt element (META tags), for example.

This view can provide benefits such as a quick assessment for relevance (for example, known author names, institution or keyword), further filtering of results and selection of relevant ones based on the exposed terms, and further reminders of more appropriate (or inappropriate) search terms that could or could not be used in further searches. The index terms could also serve as a summary to the results that could be quickly browsed in lieu of checking individual search results page-by-page.

Categorizing web search results for query refinement has also been demonstrated by Käkki (2005). Their results showed that categories help when search engine ranking fail by enabling the user to access search results that are located far down in the rank order

¹²<http://www.google.co.uk>

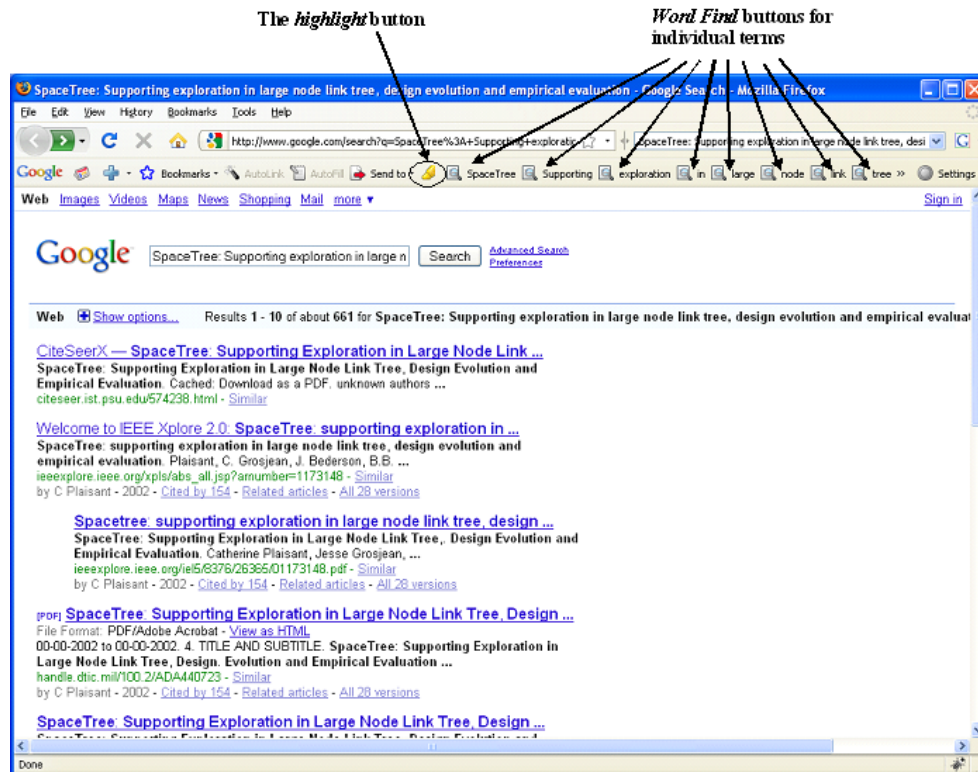


FIGURE 8.1: Presentation of Google results for a search phrase

list. In addition the user could be helped to discover information with the help of the categories and through exploration. The test users regularly used the categories and found and selected those search results that were ranked low in the search engine list and which people do not usually view as they typically only view the first results page (10 documents or less viewed in most cases) (Jansen and Pooch, 2001).

8.2.4 Evaluation Through Undirected Browsing and Extended Use

The research was conducted under limitations in terms of time and manpower. Evaluation of DeFiBro has so far been only with preset tasks using a hierarchy unknown to the user. While this helped evaluate the efficiency and effectiveness of the system to some extent it fell short of validating the system with regard to its usefulness in real settings, that is, using a real user hierarchy and not constrained with tasks. This can be achieved by installation in users' system with assurance and monitoring of use over a long period of time, followed by a survey to evaluate usefulness. Such a study will however require the system to be formally, thoroughly tested for errors before dispatching it to user desktops.

Such testing and evaluation will inevitably result in iterative, rigorous revisions based on user feedback and needs. A more reliable and well-specified recommendation, or even tool, could therefore result from this.

8.2.5 Augmentation of Hierarchies with Context, Detail and Relations

Current hierarchy navigation on the desktop could be enhanced by integration of the documents' indexes into the file system hierarchy visualization interface, provision of details such as previews and relations between documents. This will help provide context during browsing while the details and relations will provide more informed browsing which will help ease user memory load and stimulate discovery. Integration of documents' indexes could take the form of suggestions of commonly-occurring terms in the set being surveyed which could be selected to either filter the current collection or change the focus to a grouping based on the selected term.

The derivation or acquisition of indexes can also be investigated to find the best method that can be used. Fulltext analysis could be undertaken to further enhance the indexes and to influence the measurement of similarity or categorization of the documents in the current setup. Alternatively a bigger study could be undertaken to evaluate user specification of metadata against automatic system extraction of the same, and how both support retrieval and discovery.

8.2.6 Scalability and Seamless Integration with Hierarchy

Currently metadata collection and structuring is done "offline" using a separate module. This means that any updates and additions that take place on the desktop are not effected unless the module and DeFiBro are rerun manually. Direct connections between the metadata module and DeFiBro and between the module and the operating system will ensure that updates are effected automatically and the parser works on realtime data every time DeFiBro starts.

Scalability of the system depends on the speed of the parser and storage available on the user's system. Parsing the metadata is done on initialization of DeFiBro, requiring a short wait on starting the system. Currently it takes slightly less than a minute for the system to load for 1 MB (for about 2800 documents) using a Pentium 1.86 GHz processor with 512MB RAM. From the user study conducted in this research with 25 users, the average number of pdf, Word, Excel, Powerpoint presentations for a user was 4209 while the average number of all files was 39102 (in 2839 folders), indicating a need to consider accommodating a great expansion to data handled by the system. The parser's speed may be less of a problem if the processing is done at a convenient time, for example, it could be done while the system is ideal after startup in a similar manner to indexing as done by search engines. The capacity of the storage structures used by the program, on the other hand, will depend on the platform being used, but limits are more likely to be defined by the maximum memory available in the user's machine.

8.2.7 Improving User Control and Reasoning

Currently the interface only allows for browsing without the user specifying or influencing the organization of the information. This could be counteracted by the provision of a ‘search’ box for auto-selection of index terms as the user types, and by allowing selection and combination of terms into strings such that documents can be grouped and presented according to user needs. To do this the system needs to have some rules on which reasoning can be based for automatically determining the most appropriate items to satisfy the queries.

8.3 Final Remarks

This thesis has described the research undertaken to establish a way to support users of personal computers in accessing and utilizing documents stored and organized in the hierarchical file system. Using linked indexes derived from the metadata is proposed as a first step in formulating a method to augment navigation of these hierarchies to improve retrieval and discovery on the desktop. Presentation of similar items and previews have proved useful in other interfaces and have also proved useful for documents in our solution. The next step would be to establish how to incorporate the solution into existing operating system methods. This would involve design, extensive testing and evaluation coupled with iterative revisions based on user feedback. Coming up with a different tool separate from the well-established, built-in and well-supported structures that come with their systems and that the users are used to might not be useful. It is therefore anticipated it would be difficult to convince users to adopt a separate tool. This and other envisioned directions have been discussed above.

The research has provided an initial analysis and design to the wider fields of information organization, access and integration in the personal context. These form the basis of knowledge creation on the desktop which is part of the wider field of knowledge management. While this might be easier to study under the context of business and organizational systems where the data belongs to a specific domain, the variation of data in personal stores makes it more challenging in the personal domain.

Appendix A

Preliminary Study Information and Questionnaire

Consent Information

This study aims to understand the way users store and access their information in the hierarchy of directories and folders on the desktop. The first part of the study uses a script to collect data about the files, folders and sub folders on the user's system. Please note that none of your data is taken away from the system, only the enumeration of documents, how many sub folders the file/document is nested in, and the total number of folders on the system. You will have access to the script file and the data collected before it is taken away. Your data and responses are also not linked to the consent agreement below.

The questionnaire is used to obtain your perceptions on the use of metadata, browsing documents for purposes of finding and discovery of data, and organization, extracting and transfer of documents.

Please note that you have the right to stop at any time during your participation in the study.

Taking Part

By taking part in this study you are doing so at your own choice.

Participant

Witness

Participant ID_____

Name_____

Signature_____

Signature_____

Date_____

Date_____

Questionnaire –

Organizing and Browsing Documents on the Desktop

This brief questionnaire enquires about the way documents are organized, browsed and extracted on the desktop, and how sufficiently the current operating systems tools support this. I will be grateful if you could complete it. It will take only a few minutes to complete and will help establish the needs of users and how these could be addressed. The information provided will be confidential and anonymous and will not be used for any purpose other than my thesis. Thank you. *Any questions, comments or concerns should be forwarded to the researcher:*

Researcher Name: Gontlafetse Mosweunyane
 School of Electronics and Computer Science, University of Southampton
 email: gm05r@ecs.soton.ac.uk; tel: 023 8059 7688
 Supervisor Name: Dr L. A. Carr
 email: lac@ecs.soton.ac.uk; tel: 023 8059 4479

Please tick **✓** or Cross **X** in the boxes where applicable.

Browsing and Metadata

1. How often do you add or edit in-built document properties, e.g. title, subject, keywords, to documents? (This can be done through, for example, selecting “File” then “Properties” on the application menu when a document is open, or through right-clicking on a document icon and selecting “properties” from the pop-up menu in Windows.)

Never ☐ Rarely ☐ Sometimes ☐ Often ☐ Always ☐

2. Do you utilize the metadata (document properties) for the following for documents? [Tick all that apply]

Browsing/locating ☐ Organize ☐ Search ☐ View while inspecting ☐

3. (a) Do you sometimes need an overall view of all the documents you have? Yes ☐ No ☐

(b) Is this feature provided by the operating system? Yes ☐ No ☐

(c) If your answer to (b) is “Yes” please explain this feature and how you use it.

(d) If your answer to (b) is “No” how would you try to get the overall view of documents using the currently available tools provided by the operating system?

4. What other features/improvements are needed for browsing documents that are not provided by the current system?

Organization and Extraction

5. How much do the current tools sufficiently support selection, grouping and extracting of documents across several folders for transfer elsewhere?

Very much ☐ Moderately ☐ Just a little ☐ Not at all ☐

6. Is it easy to recognise and view or work with similar documents across folders?

Yes ☐ No ☐

7. What other problems do you experience during organization, grouping and transfer of desktop documents, especially if the documents are stored in many different folders?

Respondent Information

Operating System used

Microsoft Windows ☐ Macintosh ☐ Linux/Unix ☐

Occupation

Undergraduate Student ☐ Postgraduate Student ☐ Researcher ☐ Academic ☐

Clerical ☐ Other (please specify) _____

Educational Background (Degree, e.g. Computer Science)

Age distribution

18-25 ☐ 26-33 ☐ 34-41 ☐ 42+ ☐

Appendix B

The File Ontology

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE owl [
  <!ENTITY dc "http://purl.org/dc/elements/1.1/">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY owl "http://www.w3.org/2002/07/owl#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">
]>
<rdf:RDF
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/MetaOnto#"
>
  <owl:Ontology rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/MetaOnto">
    <owl:versionInfo>1.0</owl:versionInfo>
    <rdfs:label>MetadataOntology</rdfs:label>
    <rdfs:comment>Additional properties for file metadata</rdfs:comment>
  </owl:Ontology>
  <owl:Class rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/MetaOnto#File">
    <rdfs:label>File</rdfs:label>
  </owl:Class>
  <owl:Class rdf:about="http://www.w3.org/2002/07/owl#Thing">
  </owl:Class>

  <owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
```

```

MetaOnto#Author">
  <rdfs:label>Author</rdfs:label>
  <rdfs:comment>Author of the document</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Category">
  <rdfs:comment>The category the document could be placed in. This property provided
    by the user</rdfs:comment>
  <rdfs:label>Category</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Company">
  <rdfs:label>Company</rdfs:label>
  <rdfs:comment>Company the Author of the document belongs to</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Keywords">
  <rdfs:label>Keywords</rdfs:label>
  <rdfs:comment>Keywords describing what the document is about. This property is
    provided by user</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Manager">
  <rdfs:label>Manager</rdfs:label>
  <rdfs:comment>Author's Manager. This property added by user</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Security">
  <rdfs:label>Security</rdfs:label>
  <rdfs:comment>Security attributes of the document</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#Template">
  <rdfs:label>Template</rdfs:label>
  <rdfs:comment>Template used to create the document</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#application-name">
  <rdfs:comment>Name of application used to create the document</rdfs:comment>
  <rdfs:label>application-name</rdfs:label>
</owl:DatatypeProperty>

```

```
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#file-name">
  <rdfs:label>file-name</rdfs:label>
  <rdfs:comment>short name of the file(name and extension)</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#hyperlink-base">
  <rdfs:comment>This property provided by user</rdfs:comment>
  <rdfs:label>hyperlink-base</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#last-author">
  <rdfs:label>last-author</rdfs:label>
  <rdfs:comment>Last author of a document</rdfs:comment>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#last-print-date">
  <rdfs:label>last-print-date</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#last-save-time">
  <rdfs:label>last-save-time</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-bytes">
  <rdfs:label>number-of-bytes</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-characters">
  <rdfs:label>number-of-characters</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-characters-with-spaces">
  <rdfs:label>number-of-characters-with-spaces</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-lines">
  <rdfs:label>number-of-lines</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-pages">
```



```
<rdfs:label>number-of-pages</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-paragraphs">
  <rdfs:label>number-of-paragraphs</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#number-of-words">
  <rdfs:label>number-of-words</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#revision-number">
  <rdfs:label>revision-number</rdfs:label>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:about="http://www.ecs.soton.ac.uk/~gm05r/MetadataStuff/
MetaOnto#total-editing-time">
  <rdfs:label>total-editing-time</rdfs:label>
</owl:DatatypeProperty>

</rdf:RDF>
```

Appendix C

User Evaluation Tasks and Questionnaire

TASK

Introduction

The following task will help to evaluate the efficiency and utility of non-hierarchical browsing, locating, associating and grouping of files on the desktop using the system provided as opposed to browsing the file hierarchy using the tools provided by the operating system.

Your participation in this task is voluntary and you are free to stop at any time. You may also let the investigator know if a section of the task proves difficult and you can leave it and continue with the next.

Instructions

- You may browse using either Windows Explorer or the simple zoomable visualization offered by windows (clicking on folders to expand and view files and folders contained therein). Please use the search facility only as a last option for this task.
- You may change the view of files in Windows Explorer or the zoomable interface.
- You are allowed 5 minutes to study the test hierarchy provided and you may ask questions about what is stored in the folders if the folder name is not clear.
- For the tasks given use the windows visualization for browsing files and folders on the desktop, and then the system (DeFiBro) provided.
- It would help the investigator a lot if you also verbalize (say out) your thoughts as you look for things.

Task H1

1. Locate these files and point them out to the investigator:

- (a) A file named “**glen eyre halls complex residents responsibilities.doc**”
- (b) A Korea visa application form used to apply for a visa to attend a conference in 2007
- (c) A file named “**january 2009.doc**” that contains details of my research progress and schedule since January.
- (d) A file named “**Consent form and questionnaire.doc**” that I used for part of a user study for my research project.

TASK H2

2. Collect together these files and present them in group for working on at the same time:

- (a) **Files** that had to do with a *poster written for the open repositories conference that was held in April 2008.*

Please take a few minutes to answer the questions below in relation to the system (DeFiBro).

- (i) DeFiBro helps in finding documents whose location is not known

Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

- (ii) DeFiBro helps in browsing and making sense of the collection of files

Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

- (iii) The presentation of documents and items (other documents, previews, attributes) related to them helps in browsing documents

Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

- (iv) Provision of a way to collect, view and work with selected documents together is useful

Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree ☐

(c) What do you like about the interface?

(d) What do you feel could be improved?

Bibliography

- J. Aitchison, A. Gilchrist, and D. Bawden. *Thesaurus Construction and Use: A Practical Manual*. London: Aslib IMI, 2000.
- R. Akscyn, D. McCracken, and E. Yoder. KMS: A Distributed Hypermedia System for Managing Knowledge in Organisations. In *Proceedings of the ACM Conference on Hypertext, Chapel Hill, North Carolina, United States*, pages 1–20, 1987.
- J. Allinson, S. François, and S. Lewis. SWORD: Simple Web-Service Offering Repository Deposit. *Ariadne*, (Issue 54), 2008. <http://www.ariadne.ac.uk/issue54/allinson-et-al/> last accessed 28/09/09.
- Apple Inc. Mac OS X Leopard. <http://www.apple.com/macosx/>. last accessed 13/08/09.
- D. Barreau and B. Nardi. Finding and Reminding: File Organization from the Desktop. *SIGCHI Bulletin*, 27(3):39–43, 1995.
- M. J. Bates. Speculations on Browsing, Directed Searching, and Linking in Relation to the Bradford Distribution. In H. Bruce, R. Fidel, P. Ingwersen, and P. Vakkari, editors, *Proceedings of the Fourth International Conference on Conceptions of Library and Information Science (COLIS 4)*, pages 137–150. Libraries Unlimited, 2002.
- M. J. Bates. What is Browsing Really? A Model Drawing from Behavioural Science Research. *Information Research*, 12(4):330, 2007. <http://informationr.net/ir/12-4/paper330.html> last accessed 20/08/09.
- D. Beard and J.Q. Walker. Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces. *Behaviour and Information Technology*, 9(6):451–466, 1990.
- B.B. Bederson and J.D. Hollan. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. *Proceedings of the 7th annual ACM symposium on User Interface Software and Technology*, pages 17–26, 1994.
- N. Belkin. Interaction with Texts: Information Retrieval as Information-Seeking Behavior. In *Information Retrieval '93*, pages 55–66, 1993.

- T. Berners-Lee. Semantic Web Road map. <http://www.w3.org/DesignIssues/Semantic.html> Last accessed 15/09/07, 1998.
- T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In *Proceedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI06) Workshop, Athens, Georgia, 2006*.
- T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. ScientificAmerican.com, <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21> last accessed 16/10/07, 2001.
- T. Berners-Lee, L. Masinter, and M. McCahill (Editors). Uniform Resource Locators (URL). <http://www.ietf.org/rfc/rfc1738.txt> last accessed 07/04/07, December 1994.
- A. Bernstein, E. Kaufmann, C. Bürki, and M. Klein. *Wirtschaftsinformatik 2005*, chapter How Similar Is It? Towards Personalized Similarity Measures in Ontologies, pages 1347–1366. Physica-Verlag HD, 2005.
- A. Boshier. Windows Management Instrumentation: A Simple, Powerful Tool for Scripting Windows Management. *MSDN Magazine*, April 2000. <http://msdn.microsoft.com/en-us/magazine/cc302338.aspx> last accessed 29/09/09.
- S. Brin and L. Page. The Anatomy of a Large Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.
- P. Bruza. Hyperindices: A Novel Aid for Searching in Hypermedia. In *Proceedings of the ACM European Conference on Hypertext '90 (ECHT '90), Versailles, France*, pages 109–122, 1990.
- P. Bruza and T. van der Weide. Two Level Hypermedia: An Improved Architecture for Hypertext. In *Proceedings of the Database and Expert System Applications Conference (DEXA '90). (A.Tjoa and R. Wagner - Editors)*, pages 76–83. Springer-Verlag, 1990.
- V. Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, 1945.
- Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan. Personal Information Management with SEMEX. In *SIGMOD 2005, Baltimore, Maryland, USA*, pages 921–923, 2005.
- R. Cailliau and H. Ashman. Hypertext in the Web - a History. *ACM Computing Surveys (CSUR)*, 31(4es), 1999.
- D. Canter, R. Rivers, and G. Storrs. Characterizing User Navigation Through Complex Data Structures. *Behaviour and Information Technology*, 4(2):93 – 102, 1985.

- R. Capra, G. Marchionini, J. S. Oh, F. Stutzman, and Y. Zhang. Effects of Structure and Interaction Style on Distinct Search Tasks. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 442 – 451, 2007.
- L. Carr, W. Hall, S. Bechhofer, and C. Goble. Conceptual Linking: Ontology-based Open Hypermedia. In *Proceedings of Tenth International World Wide Web Conference, Hong Kong*, pages 334–342, 2001.
- C. Chatfield. *Statistics for Technology - A Course in Applied Statistics*. Chapman and Hall, 3rd edition, 1983.
- A. Cheyer, J. Park, and R. Giuli. IRIS: Integrate. Relate. Infer. Share. In *Semantic Desktop Workshop 2005, Galway, Ireland*, 2005.
- Cornell University Library. Moving theory into practice: Digital imaging tutorial. <http://www.library.cornell.edu/preservation/tutorial/contents.html> last accessed 28/09/09, 2003.
- Common Repository Interfaces Group CRIG Unconference, Birkbeck College. Bring the Repository to the Workflow. <http://www.flickr.com/photos/wocrig/2197503152/> last accessed 29/09/09, 2007.
- W.B. Croft and R.H. Thompson. I3R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science*, 38(6): 389–404, 1987.
- V. Cross. Tversky’s Parameterized Similarity Ratio Model: A Basis for Semantic Relatedness. In *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual Meeting of the North American*, pages 541–546, 2006.
- D. Cunliffe, C. Taylor, and D. Tudhope. Query-based Navigation in Semantically Indexed Hypermedia. In *Hypertext 97, Southampton UK*, 1997.
- E. Cutrell, D. C. Robbins, S. T. Dumais, and R. Sarin. Fast, Flexible Filtering with Phlat - Personal Search and Organization Made Easy. In *CHI 2006 Proceedings, Montreal, Quebec, Canada*, pages 261–270, 2006.
- D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference*, pages 318–329, 1992.
- Decentralized Information Group, MIT. The Tabulator. <http://www.w3.org/2005/ajar/tab> last accessed 29/09/09, 2006.
- X. Dong and A. Halevy. A Platform for Personal Information Management and Integration. In *Conference on Innovative Data Systems, Asilomar, Canada*, 2005.

- X. Dong, A. Y. Halevy, E. Nemes, S. B. Sigundsson, and P. Domingos. SEMEX: Toward On-the-Fly Personal Information Integration. In *Workshop on Information Integration on the Web*, 2004.
- P. Dourish, K. Edwards, A. LaMarca, and M. Salisbury. Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Transactions on Computer-Human Interaction*, 6(2):133–161, 1999a.
- P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D.G. Terry, and J. Thornton. Extending Document Management Systems with User-Specific Active Properties. *ACM Transactions on Information Systems*, 18(2):140–170, 2000.
- P. Dourish, W. K. Edwards, A. LaMarca, and M. Salisbury. Using Properties for Uniform Interaction in the Presto Document System. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, pages 55 – 64, 1999b.
- S.T. Dumais and T.K Landauer. Using Examples to Describe Categories. In *Proceedings of the 1983 ACM SIGCHI Conference*, pages 112–115, 1983.
- M. Dunlop. Reflections on Mira: Interactive Evaluation in Information Retrieval. *Journal of the American Society for Information Science*, 51(14):1269 – 1274, 2000.
- A. Edmunds and A. Morris. The Problem of Information Overload in Business Organisations: A Review of the Literature. *International Journal of Information Management*, 20(1):17–28, 2000.
- D. Elswailer, I. Ruthven, and C. Jones. Dealing with Fragmented Recollection of Context in Information Management. In *In Context-Based Information Retrieval (CIR-05) Workshop in Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05)*, 2005.
- D. C. Engelbart. NLS Teleconferencing Features: The Journal, and Shared-Screen Telephoning. Stanford Research Institute, <http://www.bootstrap.org/augdocs/augment-33076.htm> last accessed 25/10/2007, 1975.
- D.C. Engelbart. Augmenting Human Intellect: A Conceptual Framework. Summary Report, SRI Project No. 3578, Stanford Research Institute (now SRI International), Menlo Park, California, 1962.
- J. Faichney and R. Gonzalez. Goldleaf Hierarchical Document Browser. In *User Interface Conference, 2001. AUIC 2001. Proceedings. Second Australasian*, pages 13–20, 2001.
- E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD*, 25(1), 1996.

- E. Freeman and D. Gelernter. *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments* (M. Czerwinski and V. Kaptelinin - Editors), chapter Beyond Lifestreams: The Inevitable Demise of the Desktop Metaphor, pages 19–48. MIT Press, 2007.
- G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The Vocabulary Problem in Human-System Communication. *Communications of the ACM*, 30(11):964–971, 1987.
- B.G. Garrido. Organising Electronic Documents: the User Perspective - A Case Study at the European Central Bank. *Records Management Journal*, 18(3):180–193, 2008.
- J. Gemmell, G. Bell, and R. Lueder. MyLifeBits: Personal Database for Everything. *Communications of the ACM*, 49(1):89–95, 2006.
- J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex Vision. In *ACM Multimedia 02*, Juan Lens Pins, France, 2002.
- K. Getz. Setting Word Document Properties the Office 2007 Way. *MSDN Magazine*, 2006. <http://msdn.microsoft.com/en-us/magazine/cc743497.aspx> last accessed 02/10/07.
- M. Golemati, A. Katifori, E. G. Giannopoulou, I. Daradimos, and C. Vassilakis. Evaluating the Significance of the Windows Explorer Visualization in Personal Information Management Browsing Tasks. In *11th International Conference Information Visualization*, pages 93–100, 2007.
- D. Gonçalves and J. A. Jorge. Now its Personal! Evaluating PIM Retrieval Tools. In *CHI 2008 Conference*, 2008.
- D. Gonçalves and J.A. Jorge. Describing Documents: What Can Users Tell Us? In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 247–249, 2004.
- T. R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.
- F.G. Halasz, T.P. Moran, and R.H. Trigg. Notecards in a Nutshell. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface, Toronto, Ontario, Canada*, pages 45–52, 1987.
- M. A. Hearst. Clustering versus Faceted Categories for Information Exploration. *Communications of the ACM*, 49(4):59–61, 2006.
- S. Henderson. How Do People Organize their Desktops? In *CHI '04 Extended Abstracts on Human Factors in Computing Systems, Vienna, Austria*, pages 1047–1048, 2004.

- S. Henderson. Genre, Task, Topic and Time: Facets of Personal Digital Document Management. In *Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction: Making CHI Natural CHINZ '05, Auckland, New Zealand*, pages 75–82, 2005.
- A. Humermann. Usage Patterns of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, 2006.
- B. Jansen and U. Pooch. A Review of Web Searching Studies and a Framework for Future Research. *Journal of the American Society for Information Science and Technology*, 52(3):235–246, 2001.
- B. Jepson and E. Rothman. *Mac OS X Tiger for Unix Geeks*. O'Reilly Media, Inc., 2005.
- B. Johnson. TreeViz: Treemap Visualization of Hierarchically Structured Information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 369 – 370, 1992.
- S. Jones and G. W. Paynter. Automatic Extraction of Document Keyphrases for Use in Digital Libraries: Evaluation and Applications. *Journal of the American Society for Information Science and Technology*, 53(8):653–677, 2002.
- W. Jones, A. J. Phuwanartnurak, R. Gill, and H. Bruce. Don't Take My Folders Away!: Organizing Personal Information To Get Things Done. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1505–1508, 2005.
- M. Käki. Search Result Categories Help Users When Document Rankings Fail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 131–140, 2005.
- A. Kao, L. Quach, S. Poteet, and S. Woods. User Assisted Text Classification and Knowledge Management. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management CIKM '03, New Orleans, LA, USA*, pages 524 – 527, 2003.
- V. Kaptelinin and M. Czerwinski. *Beyond the Desktop Metaphor: Designing Integrated Digital Work Environments*(M. Czerwinski and V. Kaptelinin - Editors), chapter Introduction: The Desktop Metaphor and New Uses of Technology, pages 19–48. MIT Press, 2007.
- D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A Customisable General-Purpose Information Management Tool for End Users of Semistructured Data. In *Proceedings of the 2003 CIDR Conference*, 2003.
- D. R. Karger and W. Jones. Data Unification in Personal Information Management. *Communications of the ACM*, 49(1):77–82, 2006.

- B. Kelly. Using Metadata to Improve Local Searching. Exploit Interactive, Issue 5, <http://www.exploit-lib.org/issue5/metadata/> last accessed 03/04/09, 2000.
- D. Kelly. Evaluating Personal Information Management Behaviors and Tools. *Communications of the ACM*, 49(1):84–86, 2006.
- L. Kelly. Context and Linking in Retrieval from Personal Digital Archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 899–899, 2008.
- G. Kobilarov and I. Dickinson. Humboldt: Exploring Linked Data. In *Workshop (Linked Data On the Web 2008) at WWW2008*, April 2008.
- J. Koren, Y. Zhang, and X. Liu. Personalized Interactive Faceted Search. In *Proceeding of the 17th International Conference on World Wide Web, Beijing, China*, pages 477–486, 2008.
- B. Kules and B. Shneiderman. Designing a Metadata-Driven Visual Information Browser for Federal Statistics. In *Proceedings of the ACM 2003 Annual National Conference on Digital Government Research*, pages 1–6, 2003.
- W. Kules, M.L. Wilson, m.c. schraefel, and B. Shneiderman. From Keyword Search to Exploration: How Result Visualization Aids Discovery on the Web. Technical Report 1516920080208, School of Electronics and Computer Science, University of Southampton, 2008.
- B. H. Kwasnik. *Engineering for Human-Computer Interaction (J. Larson and C. Unger - Editors)*, chapter A Descriptive Study of the Functional Components of Browsing, pages 191–203. Elsevier Science, 1992.
- J. Lamping and R. Rao. The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages and Computing*, 7(1): 33–55, 1996.
- J. Lamping, R. Rao, and P. Pirolli. A Focus + Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Conference Proceedings on Human Factors in Computing Systems*, pages 401–408, 1995.
- M. W. Lansdale. The Psychology of Personal Information Management. *Applied Ergonomics*, 19(1):55–66, 1988.
- S. Li. A Semantic Vector Retrieval Model for Desktop Documents. *Journal of Software Engineering and Applications*, 2(1):55–59, 2009.
- P. Lucas and L. Schneider. Workspace: A Scriptable Document Management Environment. In *ACM CH1'94 Conference Companion*, 1994.

- T. W. Malone. How do People Organize their Desks? Implications for the Design of Office Information Systems. *ACM Transactions on Office Systems*, 1(1):99–112, 1983.
- R. Mander, G. Salomon, and Y. Y. Wong. A 'Pile' Metaphor for Supporting Casual Organization of Information. In *ACM CHI '92*, pages 627–634, 1992.
- G. Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, 1995.
- G. Marchionini and B. Shneiderman. Finding Facts vs Browsing Knowledge in Hypertext Systems. *Computer*, 21(1):70–80, 1988.
- Microsoft Corporation. Windows Vista - The Features - Search and Organization. <http://www.microsoft.com/windowsvista/features/default.mspx> last accessed 12/08/09, 2006.
- Microsoft Corporation. TTEST. <http://office.microsoft.com/en-us/excel/HP052093251033.aspx> last accessed 03/06/09, 2009a.
- Microsoft Corporation. WMI Scripting Primer: Part 1. <http://msdn.microsoft.com/en-us/library/ms974579.aspx> last accessed 30/09/09, 2009b.
- E. Miller. An Introduction to the Resource Description Framework. <http://www.dlib.org/dlib/may98/miller/05miller.html> last accessed 18/11/07, 1998.
- G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: An On-Line Lexical Database. *International Journal of Lexicography*, 3(4): 235–244, 1990.
- MINDSWAP Research Group, University of Maryland. SWOOP - A Hypermedia-Based Featherweight OWL Ontology Editor. <http://www.mindswap.org/2004/SWOOP/> last accessed 05/09/07, 2004.
- T. H. Nelson. A File Structure for the Complex, the Changing and the Indeterminate. In *Association for Computing Machinery Proceedings of the 20th National Conference, Cleveland, Ohio*, pages 84–100, 1965.
- J. Nielsen. *Usability Engineering*. Academic Press, 1993.
- National Information Standards Organization NISO Press. Understanding Metadata. <http://www.niso.org/standards/resources/UnderstandingMetadata.pdf> last accessed 02/03/09, 2004.
- R. N. Oddy. Information Retrieval Through Man-Machine Dialogue. *Journal of Documentation*, 33(1):1–14, 1977.
- G. Oleksik, M. L. Wilson, C. Tashman, E. M. Rodrigues, G. Kazai, G. Smyth, N. Milic-Frayling, and R. Jones. Lightweight Tagging Expands Information and Activity Management Practices. In *ACM CHI 2009*, pages 279–288, 2009.

- E. Oren. An Overview of Information Management and Knowledge Work Studies: Lessons for the Semantic Desktop. In *Proceedings of the 2nd Semantic Desktop and Social Semantic Collaboration Workshop, Workshop at 5th International Semantic Web Conference*, 2006.
- C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proceedings of the IEEE Symposium on Information Visualization 2002 (InfoVis02)*, pages 57–64, 2002.
- E. Rasmussen. *Information Retrieval: Data Structures and Algorithms*, chapter Clustering Algorithms, pages 419 – 442. Prentice-Hall, Inc., 1992.
- P. Ravasio, G. Schar, and H. Krueger. In Pursuit of Desktop Evolution: User Problems and Practices with Modern Desktop Systems. *ACM Transactions on Computer-Human Interaction*, 11(2):156–180, 2004.
- P. Ravasio and V. Tschertter. *Beyond the Desktop Metaphor - Designing Integrated Work Environments (V. Kaptelinin and M. Czerwinski - Editors)*, chapter Users’ Theories of the Desktop Metaphor, or Why We Should Seek Metaphor-Free Interfaces, pages 265–294. MIT Press, 2007.
- J. Rekimoto. Timescape: A time machine for the desktop environment. In *CHI 99 Extended Abstracts on Human Factors in Computing Systems*, pages 180–181, 1999.
- R. E. Rice, M. McCreddie, and S.L. Chang. *Accessing and Browsing Information and Communication*. Cambridge, MA: MIT Press, 2001.
- G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich. Data Mountain: Using Spatial Memory for Document Management. In *ACM UIST’98*, pages 153–162, 1998.
- B. Rothrock, B. A. Myers, and S. H. Wang. Unified Associative Information Storage and Retrieval. In *CHI 2006, Montreal, Canada*, pages 1271–1276, 2006.
- G. Salton. *Automatic Text Processing, The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Publishing Company, NY, 1989.
- G. Salton, A. Wong, and C.S Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11), 1975.
- L. Sauermann, G. A. Grimmes, M. Kiesel, C. Fluit, H. Maus, D. Heim, D. Nadeem, B. Horak, and A. Dengel. Semantic desktop 2.0: The gnowsiss experience. In *5th International Semantic Web Conference, Athens, GA, USA*, 2006.
- L. Sauermann and S. Schwarz. Introducing the Gnowsiss Semantic Desktop. *Proceedings of the International Semantic Web Conference 2004*, 2004.

- B. Schneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, 1996.
- m. c. schraefel, M. L. Wilson, A. Russell, and D.A. Smith. mSpace: Improving Information Access to Multimedia Domains with Multimodal Exploratory Search. *Communication of the ACM*, 49(4):47–49, 2006.
- Semantic Desktop Organization. Semantic Desktop Organization. www.semanticdesktop.org last accessed 28/08/09, 2007.
- A. Singhal. Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.
- N. Soken, B. Reinhart, P. Vora, and S. Metz. *Methods for Evaluating Usability*. Honeywell, 1993.
- Southampton ECS. <http://id.ecs.soton.ac.uk/docs/> last accessed 01/10/09, 2007.
- B. Spell. Java API for WordNet searching (JAWS). <http://lyle.smu.edu/~tspell/jaws/index.html> last accessed 21/07/08, 2008.
- J. Stasko, R. Catrambone, M. Guzdial, and K. McDonald. An Evaluation of Space-Filling Information Visualizations for Depicting Hierarchical Structures. *International Journal of Human-Computer Studies*, 53:663–694, 2000.
- L. Sweeney. *Confidentiality, Disclosure, and Data Access: Theory and Practical Applications for Statistical Agencies*, L. Zayatz, P. Doyle, J. Theeuwes and J. Lane (Editors), chapter Information Explosion. Urban Institute, Washington, DC, 2001.
- E. Swierk, E. Kiciman, N. C. Williams, T. Fukushima, H. Yoshida, V. Laviano, and M. Baker. The Roma Personal Metadata Service. *Mobile Networks and Applications*, 7(5):407–418, 2002.
- A. G. Taylor. *Introduction to Cataloging and Classification*. Libraries Unlimited, Inc., eighth edition, 1992.
- J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The Perfect Search Engine is Not Enough: A Study of Orienteering Behavior in Directed Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 415 – 422, 2004.
- W. M.K. Trochim. Research Methods Knowledge Base. Web Center for Social Research Methods http://www.socialresearchmethods.net/kb/stat_t.php last accessed 21/09/08, 2006.
- D. Tudhope and C. Binding. Faceted thesauri. *Axiomathes*, 18(2):211–222, June 2008.

- D. Tudhope and M. L. Nielsen. Introduction to Knowledge Organization Systems and Services. *New Review of Hypermedia and Multimedia*, 12(1):3–9, 2006.
- D. Tudhope and C. Taylor. Navigation via Similarity: Automatic Linking Based on Semantic Closeness. *Information Processing and Management*, 33(2):233–242, 1997.
- D. Turo and B. Johnson. Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation. In *Proceedings of the IEEE Visualization '92*, pages 124–131, 1992.
- A. Tversky. Features of Similarity. *Psychological Review*, 84(4):327–352, 1977.
- S. Volda and S. Greenberg. WikiFolders: Augmenting the Display of Folders to Better Convey the Meaning of Files. In *CHI 2009*, pages 1679–1682, 2009.
- S. Walters and F. Jayakanth. Hierarchical Browsing Interface: A Case Study with India Business Insight Database. In *Proceedings of the Workshop on Multimedia and Internet Technologies, DRTC, Bangalore*, 2001.
- L. Will. Precoordinate Indexing - Willpower Information. <http://www.info-arch.org/lists/sigia-1/0412/0003.html>, <http://www.willpowerinfo.co.uk/glossary.htm>, 2004. [last accessed 02/12/2009].
- R.M. Wilson and R.D. Bergeron. Dynamic Hierarchy Specification and Visualization. In *Information Visualization, 1999. (Info Vis '99) Proceedings. IEEE Symposium on*, 1999.
- S. Wilson and K. Potat. FeedForward: A Personal Information Application, Final Report. Technical report, JISC, August 2009. available at: <http://ie-repository.jisc.ac.uk/388/> last accessed 15/10/2009.
- Xerox Parc. Placeless Documents Project. <http://www2.parc.com/csl/projects/placeless/> last accessed 09/07/09, 1999. Xerox Palo Alto Research Center.
- N. Yankelovich, B.J. Haan, N.K. Meyrowitz, and S.M. Drucker. Intermedia: The Concept and the Construction of a Seamless Information Environment. *Computer*, 21(1): 81–86, 1988.
- K. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted Metadata for Image Search and Browsing. In *Proceedings of the ACM Conference on Computer-Human Interaction*, 2003.