

Towards ontology-based semantic file systems

Ba-Hung Ngo^{1,2}, Christian Bac², Frédérique Silber-Chaussumier², Quyet-Thang Le¹

Abstract— Semantic file systems enhance standard file systems with the ability of file searching based on file semantics. The users interact with semantic file systems not only by executing browsing tasks on a hierarchy of directories but also by executing retrieval tasks like the way that information retrieval systems usually do. In this paper, we characterize semantic file systems by comparing them with other information systems, such as information retrieval systems, Semantic Web and Semantic Desktop. We identify the issues in designing a semantic file system and propose an example ontology-based solution for these issues.

Index Terms—File Systems, Semantics, Ontology, Metadata

I. INTRODUCTION

The abstract goal of a file system is to store, retrieve, locate, and manipulate information [1]. Current file systems use files as information storage units and let users organize them in a *directory hierarchy*. The information stored in file systems may be accessed by browsing directories or by giving an accurate *path name* to the file. In both cases, the user has to remember information about the location of his file. This organization does not scale well to satisfy user's current needs because of the explosion of the amount of information, the multiple sources and

email:nbhung@cit.ctu.edu.vn, Christian.Bac@int-evry.fr, Frederique.Silber-Chaussumier@int-evry.fr, lqthang@cit.ctu.edu.vn

¹ College of Information Technology, Cantho University, 01 Ly Tu Trong, Cantho, Vietnam

² Institut National des Télécommunications, 09 Charles Fourier, 91011 Evry, Cedex, France

types of the information that one has to manage.

The first Semantic file system proposed by Gifford *et al.* [2] enhances traditional file system by supplying its users with two abilities. First, by classifying files based on their attributes, a file can appear in many virtual directories which represent file attributes. This gives its users more chance to find their files because there would be more than one path conducting to a file. Second, by supplying *retrieval task*, users can access files by specifying a set of file attributes. The recent researches on semantic file systems (SFS), ~~such as described in section III.A[3], [4], [5], [6]~~ continue bring into play the two above abilities. However, up to time, *SFS* have not been officially recognized as a new file system paradigm because their goals, their characteristics and their structures have not been clearly distinguished from the ones of the other information system paradigms, such as Information Retrieval (IR) Systems, Semantic Web and Semantic Desktop. Our discussion in this paper will ~~support for the forming argument for~~ ~~of~~ a new file system paradigm, the semantic file system. We discuss on semantic file system characteristics in section II2, identify the

principle issues in designing a semantic file system in section III3, and propose ~~to integrate an support for ontologies in the file system infrastructure-based example solution for to address~~ these issues in section IV4. The related works are represented in section V5. The last section is our conclusion and future works.

II. SEMANTIC FILE SYSTEM CHARACTERISTICS

This section will make semantic file system characteristics come out by regarding them in relation to other related paradigms, such as traditional file systems, retrieval information systems, Semantic Web and Semantic Desktop.

Ricardo and Berthier [7] classify the information retrieval models into different paradigms. A paradigm is characterized by two elements: methods for organizing information and methods for searching information. The traditional file systems belong to *hierarchical paradigm* and the web search engine, such as Yahoo [8], AltaVista [9], and Google [10] belong to *Boolean paradigm*. In hierarchical paradigm, files or documents are firstly organized in a hierarchical structure of classes, e.g. tree of directories, which group files covering related topics or concepts. Files might be searched later by moving up to the top or moving down to the bottom of the directory tree; this is the process of *browsing*. -On the other hand, Boolean paradigm

organizes information by associating keywords to each document, so that documents can be queried by specifying a list of keywords which the desired documents should contain; this is the process of interrogating. -The advantage of the hierarchical paradigm is that it proposes the help for navigation, e.g. by the directory names, so that users can find out their files step by step. However, when the directory size is large with many classes, the browsing task seems inefficient because it takes a lot of users' time and effort. The hierarchical paradigm does not permit the users to express or describe the files that they want to search. -The use of path name as a request for files in traditional file systems makes the request less expressive. Moreover the syntax of the request supports neither disjunctions, nor negations. Evenly, the conjunctions are also reduced the permutation, for example, the conjunctive request of /a/b is different with the one of /b/a. In reversely, the expressibility of user information need is the advantage of the interrogation in the Boolean paradigm. A user can specify in his request a list of the keywords ~~desired-that should be contained-in the~~ document content. -Additionally, the advanced searching in Boolean paradigm supports not only the conjunctions but also the disjunctions and negations of the keywords. The interrogating gets

users quickly attain ~~to~~ their searching goal. Therefore, SFS make every effort to enhance traditional file systems with the expressibility in user's file searching by supporting at the same time both browsing and interrogating.

The interrogating ~~gotten~~ in SFS is based on the principles of ~~(IR systems)~~ where each document is represented by a collection of index terms or keywords used to match user information need. ~~w~~Whereas IR systems use file content (document text or document structure) as the unique source to extract the index terms, SFS collect other relevant metadata for index terms. ~~We call~~We call the file semantic to refer the i nformations on which SFS is based to provide ~~their users with~~ file searching service. ~~is called file semantics.~~ Using this definition we can ~~We have~~ adapted the Dempsey and Heery definition of metadata [11] as follows: "File semantics is information associated with files which relieves their potential users of having to have full advance knowledge of their existence or characteristics. A user might be a program or a person". Beside the traditional file system's functions, SFS have to collect, organize and store file semantics in ~~the~~ ways that aresi appropriate ~~for their users to~~ search files by both browsing or interrogating ~~based on file semantics.~~

Another SFS objective is the *semantic sharing*. In the existiting IR systems and ~~existing~~ SFS, there is no way to share the syntax ~~and/or~~ the semantics of index terms ~~are not shared between the systems and even between~~ users of the same system. The same term ~~but~~ may be understood by the systems and the users with different meanings. On the contrary, two different terms may be ~~thought to used to as~~ representing ~~of~~ the same meaning. This is ~~also~~however one of the Semantic Web and Semantic Desktop objectives. Semantic Web aims ~~to at~~ extending current ~~w~~Web ~~to in which give to data~~ information is ~~given~~ well-defined meanings, better enabling computers and people to work in cooperation [12]. Semantic Desktop is one building block of Semantic Web, a Web for a single user that enables people and communities to directly collaborate with their peers while dramatically reducing the amount of time they spend for filtering and filling information. Leo *et al.* [13] show that semantic file system can be considered as one of the building block of Semantic Desktop. This also implies that the road map of SFS relates strictly to the one of Semantic Web and Semantic Desktop.

III. ISSUES IN DESIGNING A SEMANTIC FILE SYSTEM

Designing ~~of~~ a semantic file system is more difficult than designing ~~of~~ a traditional ~~file~~

~~systemone~~ because it ~~is~~ requires~~d~~ to employ many semantic technologies. This section identifies the issues ~~that our futurein~~ -designing of semantic file system ~~has to overcome~~.

A. ~~File semantic C~~ ~~concept~~ ~~sion of file semantics~~

To be able to search files by browsing or interrogating on file semantics, SFS have to collect file semantics. Therefore, it is necessary to determine which information associated to file is considered as file semantics. Each existing semantic file system proposes ~~eachits~~ own conception of file semantics and constructs the solution based on this specific view. File semantics in MIT-SFS [2] is ~~related to~~ properties such as *author, owner, date, type, subject, title, directory, name, category* of files, or *entities* in file content, e.g. a procedure of C program. *Nebula file object* [3] consists of a set of attributes that represents file properties. Sedar [4] uses semantic vectors extracted from file content to search files. LFS [5] distinguishes two types of file properties: *extrinsic property* and *intrinsic property*. The extrinsic properties are concepts such as *art, music, movie, port, seaside, the USA, capital*, that a user will assign manually to a file if he thinks that the file concerns to the concepts. The intrinsic properties are attributes such as *name, size, type* (of a file general), *artist, genre* (of a music file) that can be automatically

extracted from the file itself by programs called *Semantic collectors*~~...~~. ~~We believe that a~~ more general view of file semantics ~~willcan~~ help SFS to be designed more efficiently. ~~ThisSo we~~ ~~introduce a~~ general view ~~of file semantics by~~ ~~systematizing can be obtained splitting semantics~~ ~~its existence~~ into three classes: property-based semantics, content-based semantics and context-based semantics.

Property-based semantics is general, content and context independent information related to files, such as *owner, creation date, last modified date, type* of file,... SFS~~s~~ create property-based semantics for all files and share it for all applications and users. The existing SFS~~s~~ [2], [3], [4], [5] all support this type of file semantics.

Content-based semantics is information obtained by analyzing file content. Content-based semantics usually represents internal organization of file content: roles played by pieces of content data. The general role, called *text*, is used to specify all text in a document. The conventional *text-based search tools*, such as Glimpse [14], Google Desktop [15] and Beagle [16], index terms or keywords in text to provide their searching services. Text-based searching is also supported in the existing SFS [2], [4], [5]. There are others interesting roles that could be used for file searching. In MIT-SFS [2], the roles

such as *imports*, *exports*, *functions* can be used for searching objects in C program files. MP3 music files can be retrieved with roles such as *artist*, *album title*, and *year* in LFS [5]. HTML files can be searched for the following characteristics: having *heading* containing words "Album" and "Paris", containing "JPEG" images taken by a specific *camera*. In the last example, the heading, image and camera are the roles played by content data. To represent valid roles in file structures, metadata whose syntax and semantics are defined by applications is usually used.

Context-based semantics is the information about the interrelated conditions in which a file exists. Context-based semantics expresses relationships of a file with other subjects: other files, concepts, persons... It can be the context in the computer world, such as concurrently accessed files, the user current task, any action that the user associates with the file use [17]. Context can be connected to the real world: user's opinion on mp3 music files (exciting, normal or boring) [5], user's classification of JPEG files (family, friend, summer, vacation, wedding, work ...). Context-based semantics can be obtained from SFS and —application environment where files are handled. For instance, when a file is attached to an *email*, its

context consists of the *sender*, the *receiver* and the *subject* of the email. At the receiver side, if the file is extracted from the email and saved normally, it loses this context. However, if supported, the file will be saved in SFS together with its context-based semantics so that the file can be searched later by the sender, receiver or subject of the email that it was attached.

The existence of file semantics is determined by *agents*, such as traditional file systems, applications and users that handle the files. For instance, standard file systems usually create file metadata that can be considered as property-based semantics for all files; applications can define their own file structures that determine content-based semantics; application execution context contains contexts for files handled by the application; users via shell applications can assign their opinions to files. We use the notion *conception of file semantics* to specify all type of file semantic data that can be created and associated to files handled by an agent. Each agent has its own concept of file semantics. To be able to collect file semantics for SFS, one has to understand the corresponding conception of file semantics.

B.File semantic representability

This issue requires that collected file semantics stored in semantic store still remains its original

meaning as the ~~conception~~ of the agent who created it. This implies that SFS have to use methods for file semantic representation. In IR systems, the meaning of index terms or keywords is interpreted by users. IR systems are based on pattern-matching principle. They do not have any responsibility in assuring that the same term appearing in user query and in the searching result has the same meaning. SFS overcome this problem by integrating a mechanism that controls the meanings of the terms in SFS. File semantic representability is an essential characteristic needing to be supported in SFS.

C. File semantic extensibility

This issue requires that new ~~conception~~ of file semantics can be added to SFS easily without modifying or reconfiguring the structure of SFS. In the first step, SFS could integrate some built-in ~~conception~~ of file semantics such as property-based semantics, content-based semantics of some popular file types or context-based semantics for popular applications. Next, to provide file semantics of new applications, SFS should extend predefined semantics. So SFS have to be designed ~~with~~ taking into account file semantic extensibility.

D. Interoperability

This issue requires that file semantics created by an application, once collected by SFS, can be

understood and used by other agents without prior communication. This can be done if the ~~conception~~ of file semantics is understood by the agents using it. So SFS should ~~provide~~ have the mechanisms ~~that~~ publishes and shares a ~~conception~~ of file semantics ~~to~~ all agents in the system. The interoperability helps new ~~conception~~ of file semantics ~~in~~ being known by other applications and users.

E. File semantic standardization

This issue requires that the appearance of the same file semantics in different ~~conception~~ of file semantics has to be recognized. It means that the same term or concept ~~that~~ appears in different agents must have the same meaning. File semantic standardization makes the searching result become more relevant.

F. Combination of browsing and interrogating

As mentioned in the section 2, the goal of SFS is to support their users in file searching with both browsing and interrogating based on file semantics. It means that collected file semantics has to be organized in such a way that is appropriate for both browsing and interrogating. ~~It also means that~~ and a new query language ~~needs~~ to be defined for users to interact with SFS. A standard tree of categories, such as Yahoo [8] or Open Directory [18] can be used as a scheme for classifying ~~of~~ files automatically by

programs or manually by users. However, the large size of the tree makes users feel dizzy when browsing for files and be tired when classifying new files. MIT-SFS [2] uses a three level tree to classify automatically files according to their intrinsic file semantics. The first level will be the name of registered attributes, the second level is the values of the attributes in the first level, and the last level will be the links to the files. A path, such as */owner/smith/MyFile*, will be automatically introduced into the tree to express that the *owner* of the file *Myfile* is *smith*. Users can browse the tree step by step or make the interrogation, such as *ls -F /sfs/owner:smith/text:resume*, to find out the files whose owner is *smith* and that contain the keyword *resume*. The other SFS also permit their users to classify manually files according to extrinsic file semantics. Nebula [3] replaces traditional directories by views. A view is created to group files that share common properties, such as all text files of the semantic file system project. A view can have more than one parent view. This creates an acyclic graph of views. Users can browse views to search files or interrogates files at a view. The classification of files can be made at on-several levels. A taxonomy created from the system views will classify files for the whole system. Then, the users can

personalize this organization themselves by creating sub views from system views to satisfy their own needs. The result of an interrogation usually contains many files. LFS [5] does not rank hit files as the way that IR systems usually do. LFS aims to refine result of interrogation by classifying the hit files according to user predefined taxonomy of files properties. A user can begin with a general interrogation. Then the refine tree of result will guide him to choose a branch that leads him nearer the file he wants. Most existing SFS extend traditional file systems' name space and semantics of file commands, such as *ls*, *mkdir*, *cd*, to define a language for browsing and interrogating. The advantage of this approach is that new characteristics of SFS can be introduced to users without influencing users to continue use traditional file system. However, the hard constraint of path name limits expressibility of user request.

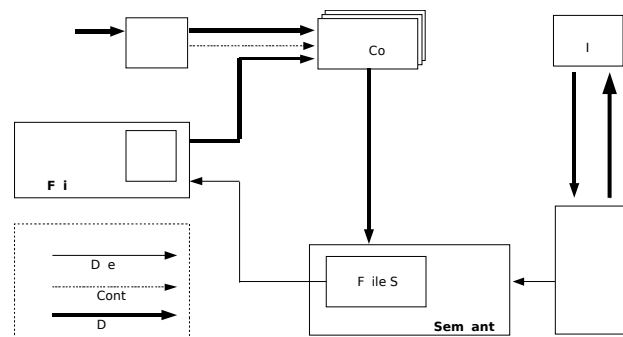


Fig. 1. Naive SFS Model

G.SFS implementation approaches

A semantic file system can be implemented by *augmented* or *integrated* approaches. Whereas

augmented approach uses traditional file systems as a file store, integrated approach create it owns file store. The survey of Vasudevan and Pazandak [19] shows that the integrated approach is the quick way to get the realization of SFS, if end-users are willing to accept more drastic changes in their operating systems. In reversely, the augmented approach provides less perfect implementations of the functionality that a user can expect from an SFS, but with almost no negative perturbation to the user's current file access capabilities. If augmented approach is chosen, some questions, such as where to store file semantic data? And which level to integrate semantic operations? have to be answered. The existing SFS [2], [3], [4], [5] use augmented approach for their implementation.

IV. INTEGRATING ~~OF~~ ONTOLOGY INTO SEMANTIC FILE SYSTEMS ~~—AN EXAMPLE SOLUTION~~

The use of file semantics in SFS to provide semantic-based searching is described in naive *Fig. 1*. File semantics is collected by many Semantic collectors. Depending on the occurred event and the type of handled file, Semantic collectors are called to collect file semantics. An agent can define its Semantic collectors to collect interesting file semantics. Usually, SFS support Semantic collectors to collect property-based semantics and content-based semantics for

popular file types. For specific or new file types, only the application designers know what content-based semantics is and how to collect it from the file content. So the application designers are responsible for defining new Semantic collectors. Similarly, there will be many Semantic collectors supported by applications to collect context-based semantics. Although collected from different sources, all file semantics is finally stored together in a Semantic Store in a structure that is appropriate for Semantic Search Engine to explore and provide users with semantic-based searching.

We found that the five first issues concerned to file semantics in section 3 can be attained if we integrate the support for ontologies into SFS. Ontology is a representation of a knowledge domain. Ontologies are mainly used for knowledge sharing and reuse across different applications. Ontology is defined as specifications of representational vocabulary for a shared domain of discourse (definitions of classes, relations, functions and other objects) [20]. Ontologies permit to share meanings of terms to overcome barriers created by disparate vocabularies, approaches, representations, and tools in a given domain. Qin and paling [21] propose the conversion of controlled vocabularies into ontologies to get deeper semantics in

describing digital objects, both conceptually and relationally. Ontology can be used in many application categories, such as common access information and ontology-based search [22]. Many ontology technologies have been developed, for example: ontology representation language (RDF [23], OWL [24], and XTM [25]), ontology search engine (JENA [26], Ontopoly [27]), query language (RDQL [28], SPARQL [29], and TMQL [30]), graphic ontology representation (Vizigator [31]), and ontology editor (Protégé [32]).

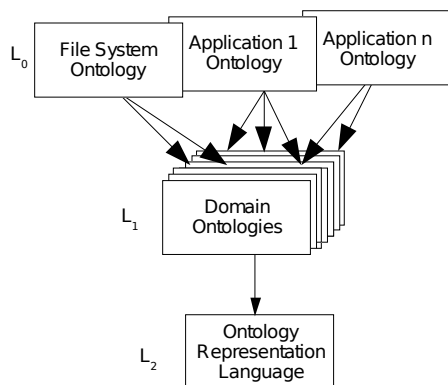


Fig. 2. Multi-ontology layer approach for SFS.

In our SFS model, ontology is used to specify conception of file semantics. For instance, SFS can use ontology to specify property-based semantics that they can create~~d~~ for files. A mail user agent can specify context-based semantics that he can save~~d~~ together with attached files by means of ontology. Usually a conception of file semantics concerns many knowledge domains and one knowledge domain may be shared by many applications. We adopt multi-ontology layer

approach proposed by Uschold and Jasper [22] and already used by Xiao and Cruz [33] to share conception of file semantics. Uschold and Jasper [22] proved that for sharing ontologies at level L_i , it's required to refer to ontologies at level L_{i+1} . We define two types of ontologies: *domain ontologies* and *application ontologies*; see Fig. 2.

Domain ontologies specify terms, concepts and their relationship for different knowledge domains. A concept is usually represented by a class having many properties. A class can be inherited from one class and can be the parent of other classes. A concept can belong to many knowledge domains. For instance, an ontology for computer science academic departments [34] can be have the terms such as *AdministrativeStaff*, *Article*, *Assistant*, *AssistantProfessor*, *AssociateProfessor*, *Book*, *Chair*, *Course*, *ConferencePaper*, *ClericalStaff*,... and the term *Book* also appears in other domain ontologies, such as the ontology that models documents, particularly publications [35]. Therefore, it's necessary to have a mechanism that permits to shares terms between different knowledge domains. The ontology representation languages, such as DAML+OIL [36] and OWL [24], are designed to answers this requirement. In addition, domain ontologies can be classified into categories, such as Open Directory [18]. The

domain ontology [34] is classified in Open Directory at the branch *Computers/Computer_Science/Academic_Departments/*.

The information handled and stored in files by an application is usually concerned to one or many knowledge domains; therefore semantics of a file is also concerned to one or many domains. As defined in the section 3III.4A, the notion of conception of file semantics is used to specify all file semantic datas that ~~is possible to becan be~~ created and associated to files handled by an agent. It is appropriate to specify ~~conception~~ of file semantics by ontology, so-called *application ontology*, that use terms and concepts defined in underlying shared domain ontologies. All domain ontologies are finally written in the same ontology representation language. Whereas ontology representation language helps SFS having the file semantic representability, the sharing domain ontologies between application ontologies helps SFS having the file semantic standardization.

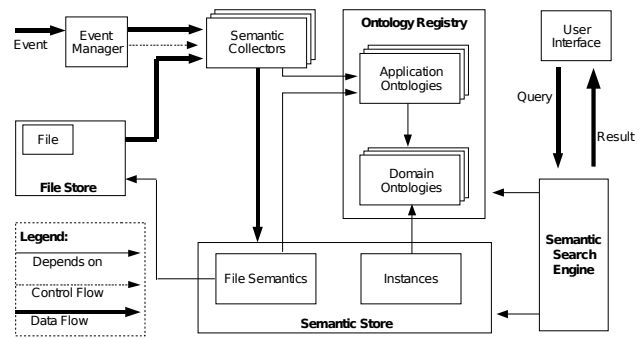


Fig. 3. Ontology-based semantic file systems.

We propose a semantic file system model that supports multi-layer ontology approach in Fig. 3. This model inserts the new Ontology registry component into the general SFS model presented at the beginning of this section. Ontology registry maintains information about the application ontologies and domain ontologies that have already been registered. A ~~conception~~ of file semantic is only available in SFS if its application ontology is registered in Ontology registry. The domain ontologies must be inserted in Ontology registry before being referenced by application ontology. In the same way, a Semantic collector has to register its application ontology before collecting semantic data. One could add new application ontologies into Ontology registry for file semantic extensibility. By browsing Ontology registry, one agent can discover the ~~conception~~ of file semantics of other agents. As a result, Ontology registry also provides interoperability.

The collecting of intrinsic file semantics and representing it by ontologies is not difficult because designer of semantic collector

understand the meaning of information - intrinsic file semantics- that he extracts from the file. For example, it is not difficult to extract and represent semantically the information about artist, album title and year of mp3 music files. However, the problem is quite different for extrinsic file semantics. User can assign to files via shell programs (or graphic user interface programs) [5], [6] any terms, such as *apple*, *driver* representing the subjects that the files talk about. It's difficult for the Semantic selector of shell programs to determine the meaning of these terms. Apple, is it a computer name or a fruit name? Driver, is it a software or a job? However, if semantic selectors know that the files are related to computer domain, the answer is quite easy. So, to create the semantic selector for shell program some technology that permits mapping terms assigned to files by user into concepts of knowledge domains should be employed.

The **using** of ontology technology for file semantic representation makes interrogating more expressive. At the low level, some query languages such as RDQL [28], SPARQL [29], and TMQL [30] can be used to request semantic search engine for files based on their semantics. At user interface level, the simple form of query, for instance Boolean query, should be used. The result of interrogating can be refined by

classification of knowledge domain and the relationship of concepts.

V.RELATED WORKS

"The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [12]. The World Wide Web Consortium [37] promotes the Semantic Web technologies to provide a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is based on RDF, which integrates a variety of applications using XML for syntax and URIs for naming. While XML is a powerful, flexible syntax for structured documents, it imposes no semantic constraints on the meaning of these documents. RDF [23] is for semantics and provides a powerful framework for supporting the exchange of knowledge on the Web. OWL [21] provides a language for defining structured, web-based ontologies which deliver richer integration and interoperability of data among descriptive communities. According to Eric Miller, W3C Semantic Web Activity Leader, "The Semantic Web is made through incremental changes, by bringing machine-readable descriptions to the data and documents already on the Web." [38]

Topic Maps published in the standard ISO/IEC1325 defines a model for the semantic structuring of knowledge networks. Topic Maps is designed to manage the information glut, build valuable information network over any kind of information resources [39]. Topic maps can be regarded as the International standard for codification that is the necessary prerequisite for the development of tools that assist in the generation and transfer of knowledge [40]. Subjects are the starting point for Topic Maps. A subject may be a concept, idea, notion or "anything whatsoever" that is worth of becoming a topic in a topic maps. A topic describes a subject by its three characteristics: topic names - human readable name for the subject, occurrences - link to information resources relevant for topic and associations with other topics. Topic maps can be used as a mechanism to control vocabulary and to represent taxonomies, thesauri, faceted classification, synonym rings and authority files [41]. The standard XTM [25] interchange format for topic maps has been standardized. A standard schema language for topic maps, called TMCL (Topic Map Constraint Language) [42] is under development. Many free topic map engines and tools have been developed [43]. OKS Samplers is an example of using topic maps for creating ontologies,

instances of ontologies and ontology-based searching [44].

Semantic Desktop deals with transferring the Semantic Web consisting of technology, philosophy and people involved to desktop computers. Sauermann *et al.* [45] define that "A Semantic Desktop is a device in which an individual stores all her digital information like documents, multimedia and messages. These are interpreted as Semantic Web resources, each is identified by a Uniform Resource Identifier (URI) and all data is accessible and queryable as RDF graph. Resources from the web can be stored and authored content can be shared with others. Ontologies allow the user to express personal mental models and form the semantic glue interconnecting information and systems. Applications respect this and store, read and communicate via ontologies and Semantic Web protocols. The Semantic Desktop is an enlarged supplement to the user's memory". Sauermann *et al.* [13] also show that ontology-based file system is one of the building blocks supporting data and information for Semantic Desktop.

VI. CONCLUSION AND FUTURE WORKS

Beside the traditional file system's functions, the main goal of SFS is to collect, organize and store file semantics in the way that is appropriate for their users to search files by both browsing or

interrogating based on file semantics. By regarding SFS in relation to other related paradigms, such as traditional file systems, IR systems, Semantic Web and Semantic Desktop, we characterised SFS. Based on existing SFS and related works, we identified three types of file semantics: property-based, content-based and context-based semantics. Controlling vocabulary is the major issue of SFS semantic file systems, i.e it would provide interoperability between SFS, applications and final users. We have proposed an ontology-based example solution where multi-layer ontology approach is employed. In the next months, we will further investigate the design of ontology-based semantic file system. In particular we want to investigate how to reuse and integrate ontology technologies such as representation languages, domain ontology libraries, semantic search engines and query languages and provide a first prototype.

REFERENCES

- [1] D. Giampaolo, Practical File System with the Be File System, Morgan Kaufmann Publishers, San Francisco, California, 1999
- [2] D. K. Gifford, P. Jouvelot, J. J. W. O'Toole, and M. A. Sheldon, "Semantic File Systems," Proceedings of the 13th ACM Symposium on Operating Systems Principles, 1991, pp 16-25
- [3] M. Baruah, C. M. Bowman, B. Camargo, C. Dharap, and S. A Potti, "File System for Information Management," Proceedings of the International Conference on Intelligent Information Management Systems, 1994
- [4] M. Mahalingam and C. Tang, Z. Xu, "Towards a Semantic, Deep Archival File System," The 9th International Workshop on Future Trends of Distributed Computing Systems, 2003
- [5] Y. Padiouleau, "Logic File System, un système de fichier basé sur la logique," Université de Rennes 1, 2005
- [6] S. Bloehdorn, O. Görlitz, S. Schenk and M. Völkel, "TagFS --- Tag Semantics for Hierarchical File Systems," Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria, September 2006
- [7] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, Addison-Wesley, 1999
- [8] Yahoo, <http://www.yahoo.com>
- [9] AltaVista, <http://www.altavista.com>
- [10] Google, <http://www.google.com>
- [11] L. Dempsey and R. Heery, "Metadata: a current view of practice and issues," Journal of Documentation, 54 (2), 1998, pp 145-172
- [12] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, 2001
- [13] A. Bernardi, A. Dengel, and L. Sauermann, "Overview and Outlook on the Semantic Desktop," Proceeding of Semantic Desktop Workshop, 2005
- [14] U. Manber and S. Wu, "Glimpse: a tool to search through entire file systems," Proceedings of the USENIX Winter Conference, 1994, pp 23-32
- [15] Google, Google Desktop. <http://desktop.google.com/>
- [16] Beagle, http://beaglewiki.org/Main_Page
- [17] G. R. Ganger and C. A. N. Soules, "Connections: Using Context to Enhance File Search," Proceedings of the 20th ACM Symposium on Operating Systems Principles, Brighton, UK, 2005, pp 119-132
- [18] Open Directory, <http://dmoz.org/>
- [19] V. Vasudevan and P. Pazandak., Semantic File Systems, *Object Services and Consulting, Inc*, 1996, <http://www.objs.com/survey/OFSExt.htm>.
- [20] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Guarino & Poli (Eds), 1993
- [21] S. Paling and J. Qin, "Converting a controlled vocabulary into an ontology: the case of GEM," Information Research, Vol. 6 No. 2, 2001
- [22] R. Jasper and M. Uschold, "A Framework for Understanding and Classifying Ontology Applications," Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR5), 1999
- [23] W3C Recommendation, RDF Primer, <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>, 2004
- [24] W3C Recommendation, OWL Web Ontology Language Overview, 2004
- [25] TopicMaps.Org, XML Topic Maps (XTM) 1.0, <http://www.topicmaps.org/xtm/1.0/> (2001)
- [26] <http://jena.sourceforge.net/>
- [27] <http://www.ontopia.net/>
- [28] <http://www.w3.org/Submission/RDQL/>
- [29] <http://www.w3.org/TR/rdf-sparql-query/>
- [30] <http://www.isotopicmaps.org/tmq1/>
- [31] <http://www.ontopia.net/solutions/vizigator.html>
- [32] <http://protege.stanford.edu/>
- [33] I.F. Cruz and H. Xiao, "A Multi-Ontology Approach for Personal Information Management," Proceeding of 1st Workshop on The Semantic Desktop, International Semantic Web Conference, Galway, Ireland, 2005
- [34] <http://www.daml.org/ontologies/64>
- [35] <http://www.daml.org/ontologies/62>
- [36] <http://www.daml.org/>
- [37] The World Wide Web Consortium, <http://www.w3.org/>
- [38] World Wide Web Consortium. Issues RDF and OWL Recommendations. <http://www.w3.org/2004/01/sws-pressrelease>.
- [39] H. H. Rath, White Paper - The Topic Maps HandBook, Empolis GmbH, 2003
- [40] S. Pepper, "The TAO of Topic Maps - Finding the Way in the Age of Infoglut", Ontopia, <http://www.ontopia.net/topicmaps/materials/tao.html>
- [41] L.M. Garshol, "Metadata? Thesauri? Taxonomies? Topic Maps! Making sense of it all," Journal of Information Science, volume 30, number 4, Chartered Institute of Library and Information Professionals, 2004, pp 378-391

- [42] ISO/IECJTC1/SC34, Topic Maps Constraint Language,
<http://www.isotopicmaps.org/tmcl/tmcl-2005-02-12.html>.
- [43] <http://www.topicmap.com/topicmap/tools.html>.
- [44] U. Manber and S. Wu, Ontopia. OKS Samplers,
<http://www.ontopia.net/download/freedownload.html>.