

# Intégrer des ontologies pour améliorer les systèmes de fichiers sémantiques

Baezung Ngo<sup>1,2</sup>, Christian Bac<sup>2</sup>, Frédérique Silber-Chaussumier<sup>2</sup> et lqthang@cit.ctu.edu.vn<sup>1</sup>

**Résumé**— Les systèmes de fichiers sémantiques améliorent la recherche d'information en introduisant des possibilités de recherche basées sur des expressions sémantiques relatives aux fichiers. Les utilisateurs interagissent avec un système de fichiers sémantique non seulement en parcourant une arborescence mais aussi en posant des questions comme dans un système de recherche d'informations. Cet article précise les points clés associés à la conception des systèmes de fichiers sémantiques et propose d'améliorer leurs possibilités en intégrant le support d'ontologies.

**Mots clés**—Systèmes de fichiers, sémantique, métadonnées, ontologies

## INTRODUCTION

Un système de fichiers permet de stocker, restaurer, localiser et manipuler des informations dont l'unité de base est le fichier[1]. Les systèmes de fichiers hiérarchiques laissent les utilisateurs organiser leurs fichiers en un ensemble de répertoires à leur convenance. Les informations d'un système de fichiers peuvent être retrouvées en parcourant l'arborescence et chaque fichier est identifié de manière unique par le chemin qui le relie à la racine de l'arborescence. Dans tous les cas, l'utilisateur doit mémoriser des informations relatives à l'emplacement du fichier. L'augmentation de l'espace de stockage, du nombre des fichiers mais aussi de la variété des types de fichiers et de la diversité de leurs origines rendent ces opérations de recherche fastidieuses. Le premier système de fichiers sémantique (SFS) proposé par Gifford *et al.* [2] améliore les systèmes de fichiers traditionnels de deux manières. Premièrement il permet à 'un fichier classifié à partir d'attributs d'apparaître dans plusieurs répertoires qui représentent chacun un type d'attribut différent. Deuxièmement, il permet de rechercher des fichiers à partir d'attributs. Les systèmes de fichiers sémantique actuels, tels que décrits dans la section II, intègrent ces deux possibilités, mélangeant ainsi l'aspect traditionnel arborescent d'un système de fichiers avec les possibilités des systèmes de recherches d'information. Les systèmes de fichiers sémantiques partagent des paradigmes et des technologies avec les systèmes de recherche d'information, le Web sémantique et les bureaux sémantiques. Cet article présente les principales caractéristiques des systèmes de fichiers sémantiques dans la section II. Il dégage les points clés de la réalisation d'un système de fichiers sémantique dans la section III. Il propose, dans la section IV, d'intégrer le support d'ontologies directement dans le système de fichiers pour répondre aux besoins exprimés dans la section précédente. La section V compare notre proposition avec les travaux dans les domaines où la sémantique est associée à la recherche d'information. Nous terminons par des conclusions et en invoquant nos futurs travaux.

## I. SYSTÈMES DE FICHIERS SÉMANTIQUES

Baeza Yates and Ribeiro-Neto [7] classifient la recherche d'information selon deux techniques la navigation (browsing) et l'interrogation (querying). Les systèmes de fichiers organisent les fichiers en un arborescence qui peut être naviguée. Les moteurs de recherche permettent de retrouver un document relativement à son contenu, le plus souvent à l'aide de mots clés. Navigation et interrogation présentent des avantages et les systèmes de fichiers sémantiques mélangent les deux types de recherche. Ces systèmes de fichiers doivent présenter les caractéristiques décrites dans les sections suivantes.

### A. Organisation des fichiers basée sur la sémantique

Dans les systèmes de fichiers sémantique les fichiers ne sont pas organisés manuellement en répertoires par les utilisateurs mais sont automatiquement organisés selon leur sémantique. Chaque système de fichiers sémantique existant a ses propres concepts de sémantique. La sémantique dans MIT-SFS [2] est liée aux propriétés des fichiers telles que l'auteur, le propriétaire, la date, le type, le sujet, le titre, l'annuaire, le nom, la catégorie, ou les entités dans le contenu du fichier, par exemple une fonction de programme C. Un objet de type fichier de Nebula [3] se compose d'un ensemble d'attributs qui représente les propriétés de fichier. Sedar [4] emploie des

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

<sup>1</sup> College of Information Technology, Cantho University, 01 Ly Tu Trong, Cantho, Vietnam

<sup>2</sup> Institut National des Télécommunications, 09 Charles Fourier, 91011 Evry, Cedex, France

vecteurs sémantiques extraits à partir du contenu de fichier pour rechercher des fichiers. LFS [5] distingue deux types de propriétés de fichier : les propriétés extrinsèques et les propriétés intrinsèques. Les propriétés extrinsèques sont des concepts qu'un utilisateur peut assigner manuellement à un fichier. Les propriétés intrinsèques sont des attributs qui dépendent du fichier lui-même. Ces attributs peuvent être associés aux méta données du système de fichier ou peuvent être déduits du contenu du fichier. Les programmes qui permettent cette extraction automatique sont appelés des collecteurs sémantiques. En plus, Connections [17] emploie le contexte de fichier pour classer la pertinence des recherches de fichier. Nous adaptons la définition de Dempsey et Heery des métadonnées [11] comme suit : la "sémantique de fichier est l'information liée aux fichiers qui évite aux utilisateurs d'avoir une connaissance anticipée de leur existence ou caractéristiques. Un utilisateur peut être un programme ou une personne".

Les sémantiques de fichier peuvent être classifiées en trois groupes : la sémantique basée sur les propriétés, celle basée sur le contenu et celle basée sur le contexte.

La sémantique basée sur les propriétés utilise les informations relatives aux fichiers indépendamment de leur contenu et de leur contexte. Ces informations sont les métadonnées d'un système de fichiers classique telles que le propriétaire, et les dates. Les SFS existant tels que MIT-SFS [2], Nebula [3], Sedar [4], and LFS [5] supportent des sémantiques basées sur les propriétés et permettent aux applications et aux utilisateurs de partager ces informations sémantiques.

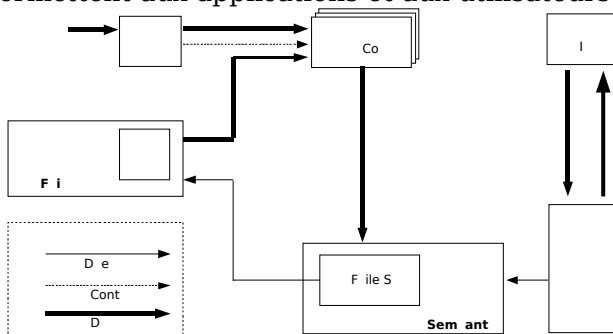


Fig. 1. SFS Model

La sémantique basée sur le contenu est relative à la structure interne des fichiers et à leurs données. Ce type de sémantique est découpé en rôle. Le rôle le plus général appelé texte est associé à l'ensemble du texte contenu dans le document. Les outils de recherche basés sur le texte tels que Glimpse[14], Google-Desktop[15] ou Beagle[16] utilisent des mots clés pour indexer les fichiers. Certains SFS permettent également la recherche basée sur le texte [2], [3], [4], [5]. D'autres rôles intéressants peuvent être utilisés pour la recherche de fichiers. Par exemple dans MIT-SFS [2], les définitions ou utilisations de fonctions en C peuvent être utilisés. De même LFS[5] permet de retrouver des fichiers musicaux à partir des informations sur les artistes, ou les titres des disques.

*La sémantique basée sur le contexte est relative aux événements systèmes qui apparaissent de manière concomitante avec l'utilisation du fichier.* Ce type de sémantique permet d'envisager des relations entre un fichier et d'autres sujets. Ce contexte peut être relatif à l'état du système ou aux actions réalisées par l'utilisateur dans une fenêtre de temps associée à la manipulation du fichier comme dans le systèmes appelé Connections [17]. Ce contexte peut aussi être associé avec une classification humaine comme dans LFS[5].

SFS collect file semantic data and maintain the relation between them and their files. This enable files to be semantically grouped. For instance, the files with the same *author* are automatically grouped together. The use of file semantics in SFS to provide semantic-based searching is described in 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.

The existence of file semantics is determined by *agents*, defined as traditional file systems, applications and users. For instance, standard file systems usually create file metadata that can be

considered as property-based semantics; applications can define their own file structures to determine content-based semantics; application execution provides the context for related files. We use the notion *concept of file semantics* to specify all types 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 concept of file semantics.

#### *B. Multi-path file accessing*

Because files are semantically classified, there will be many paths leading to a file. An mp3 music file can be access by paths such as *musics/mp3/artist*, *music/mp3/album*, or *mp3/year*. Users have more possibilities to find a file.

#### *C. Semantic-based file searching*

Searching in SFS is based on semantics with both querying and browsing. Querying enables users flexibly express their request by means of file semantics, such as in MIT-SFS [2] where *ls -F /sfs/owner:smith/text:resume* searches for files whose owner is *smith* and that contain the keyword *resume*. There usually exists some type of relationships between the concepts representing file semantics, e.g. axioms, generalization and specification relationship. These relationships enable users to browse for files or refine query result. When querying for files relating to *file system* concept, the result can also contain the files relating to *ext2fs* [5], because the SFS know that *ext2fs* is a type of *file systems*. In addition, context-based semantics such as *mail sender* can be used to search for downloaded files.

#### *D. Semantic-based file service*

Traditional file operations such as *open*, *create*, *write*, *read* and *close* are based on the concept of directory that is not exist any more in the SFS. Therefore, SFS have to port them to new context where traditional directories are replaced by file semantics.

#### *E. Semantic sharing*

Nowadays, there are many applications such as Semantic Web, Semantic Desktop, and IR systems that use semantics together with their data. Each application defines its own concept of semantics. They collect and store file semantics in different semantic stores. There is redundant in semantic storage. Information about artist, genre, and album of mp3 music is repeated in many application semantic stores. This is not efficient and prevents the interoperability between applications. It is appropriate for SFS to replace applications in the role of collecting and storing file semantics and then share file semantic for all applications.

In conclusion, beside the traditional file system's functions, SFS have to collect, organize and store file semantics in ways that are appropriate to enhance traditional file system with the five advanced characteristics: semantic-base file organization, multi-path file access, semantic-based file searching, semantic-based file service and semantic sharing.

#### Issues in designing a semantic file system

As shown in the previous section, designing a semantic file system requires many semantic technologies. This section identifies the issues in the design of semantic file system.

#### *F.*

#### *G. File semantic representability*

The file semantic representability 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 assure that same terms in user query and in the searching result have the same meaning. SFS should overcome this with a mechanism that controls the meanings of the terms in SFS. File semantic representability is essential in SFS.

#### *H. File semantic extensibility*

The file semantic extensibility requires that new concept of file semantics can be added to SFS without modifying or reconfiguring the structure of SFS. In the first step, SFS integrate some built-in concepts 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 extensibility.

### I. Interoperability

The interoperability 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 concept of file semantics is understood by the agents using it. So SFS should provide mechanisms to publish and share file semantics between all agents in the system.

### J. File semantic standardization

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

K.

L.

## II. INTEGRATING ONTOLOGY INTO SEMANTIC FILE SYSTEMS

The issues in section III can be reached through 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 enable 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]).

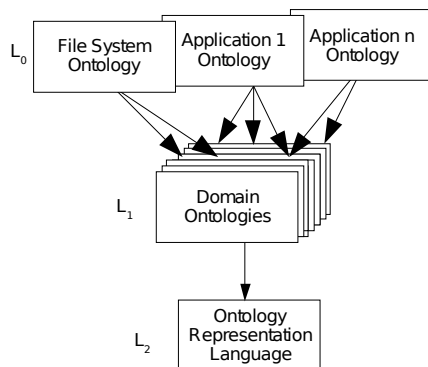


Fig. 2. Multi-ontology layer approach for SFS.

In our SFS model, ontology is used to specify concept of file semantics. For instance, SFS can use ontology to specify property-based semantics that they can create for files. A mail user agent can specify context-based semantics that he can saved together with attached files by means of ontology. Usually a concept 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 concept 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, the ontology for computer science academic departments [34]

can have the terms such as *AdministrativeStaff*, *Article*, *Assistant*, *AssistantProfessor*, *AssociateProfessor*, *Book*, *Chair*, *Course*, *ConferencePaper*, and *ClericalStaff*. 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 enables to share terms between different knowledge domains. The ontology representation languages, such as DAML+OIL [36] and OWL [24], are designed to answer 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 *Computers /Computer\_Science/ Academic\_Departments*

Information handled and stored in files by an application usually concern one or many knowledge domains; therefore semantics of a file is also related to one or many domains. As defined in the section II.A, a concept of file semantics is used to specify all file semantic data created and associated to files by an agent. It is appropriate to specify a concept of file semantics with an 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 provides SFS with the file semantic representability, sharing domain ontologies between application ontologies provides file semantic standardization.

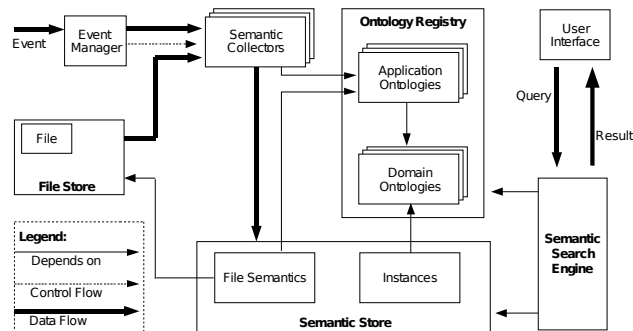


Fig. 3. Ontology-based semantic file systems.

We propose a semantic file system model that supports multi-layer ontology approach in Fig. 3. This model inserts an Ontology registry component into the general SFS model presented in the section II. The 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 the ontology registry for file semantic extensibility. By browsing the ontology registry, one agent can discover the concept of file semantics of other agents. As a result, the 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 files via shell programs (or graphic user interface programs) [5], [6] any terms, such as *apple*, *driver* related to files. 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 enables mapping terms assigned to files by user into concepts of knowledge domains should be employed.

The use of ontology technology for file semantic representation makes querying 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 querying can be refined by classification of knowledge domain and the relationship of concepts.

### III. 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]. The semantic technologies developed for Semantic Web such as Semantic Representation Language, Domain Ontologies for different knowledge domains, Semantic search engines, ... are good references for designing SFS.

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) [10] is under development. Many free topic map engines and tools have been developed [9]. OKS Samplers is an example of using topic maps for creating ontologies, instances of ontologies and ontology-based searching [8]. The ontological engineering, open source topic map software, topic map visualization can provide a foundation for SFS.

Semantic Desktop deals with transferring the Semantic Web consisting of technology, philosophy and people involved to desktop computers. Sauermann *et al.* [13] 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. SFS can share with Semantic Desktop the way in which ontological technologies are used to give meaning to information.

#### IV.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 querying based on file semantics. Based on existing SFS and related works, we identified three types of file semantics: property-based, content-based and context-based semantics. Regarding SFS in relation to other related paradigms, such as traditional file systems, IR systems, Semantic Web and Semantic Desktop, we propose that SFS should enhance traditional file system with the five advanced characteristics: semantic-base file organization, multi-path file access, semantic-based file searching, semantic-based file service and semantic sharing. 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 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. Padioleau, "Logic File System, un système de fichier basé sur la logique," Université de Rennes 1, 2005
- [6] S. Bloehdorn, O. Görnitz, 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] U. Manber and S. Wu, Ontopia. OKS Samplers, <http://www.ontopia.net/download/freedownload.html>.
- [9] <http://www.topicmap.com/topicmap/tools.html>.
- [10] ISO/IECJTC1/SC34, Topic Maps Constraint Language, <http://www.isotopicmaps.org/tmcl/tmcl-2005-02-12.html>.
- [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. Saueremann, "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](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,
- [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/tmql/>
- [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]
- [43]
- [44]