Semantic 'LS' - An Approach for Personal and Private Group Information Management

Ajay Krishnan
Indian Institute of Technology Kanpur
ajaykri@cse.iitk.ac.in

M Seetha Ramaiah
Indian Institute of Technology Kanpur
msram@cse.iitk.ac.in

Vinay Kumar Jaasti
Indian Institute of Technology Kanpur
jaasti@cse.iitk.ac.in

TV Prabhakar
Indian Institute of Technology Kanpur
tvp@cse.iitk.ac.in

ABSTRACT
One of the problems in Personal Information Management is handling the huge amount of content one seems accumulate over time on one's own personal computer related to one's domain of interest. There is however a related problem as well – that of sharing this knowledge with members of the community one belongs to. Many Knowledge Communities are typically small with substantial trust and co-operation within. Typical examples are the research groups one sees in Universities, Labs and even spanning Institutional boundaries. The problem of personal information management can be extended to sharing of information resources amongst such groups extending the PIM to a Group Information Manager (GIM). In this paper we describe an implementation of a Personal Information Manager that does the work of properly arranging the resources, not based on the physical location of the resource but based on the semantic properties of the resource, thus providing an easy to use interface to browse the resources based on their content and not just based on their physical locations. Easy to use searching interfaces based on set theory and Boolean logic are also provided so that novice users can form accurate queries for finding the desired resources easily. These resources might also be required to be shared amongst a community of users that are working on similar domain, so we convert this PIM into a GIM where not only the resources that are available in the local machine are managed but also those resources residing on remote trusted machines connected to each other in a P2P fashion.

Author Keywords
GIM, PIM, P2P, Semantics

ACM Classification Keywords
H.3 Information Storage and retrieval

INTRODUCTION

There is a lot of content on the Web and it is ever increasing. We generally download the resources from the domain of our interest and store it on our personal computers in some physical folder. But often due to the large amounts of data, retrieving the data back from the stored physical folder becomes painful as we tend to forget the place or name of the resource that was stored. We only remember what the resource was talking about or the topic of the resource. It would be helpful if we could retrieve the resources based on what they talk about rather than where they are located.

Consider the following case: A Professor teaching software architecture has a lot of resources such as papers, articles, e-books, presentations, images etc on software architecture on his personal computer. Also the Professor has some of his students who also have lots of data on software architecture stored on their personal machines. Because of the huge amount of resources available on his machine, he is unable to remember where he stored some article that talked about design patterns written by some author named X. Also he would like to share his collection of resources with his students and vice-versa.

To solve these problems the Professor needs a system by which the documents can be stored based on their content and not based on their physical locations and which is also capable of providing some services using which resources stored in remote machines are also accessible by using their contents as a reference.

Problem Statement

There is a network in which all the personal computers are connected with each other in a Peer to Peer fashion. All the machines have similar capabilities – can act as a server as well as a client. This network forms a *private group* in the sense that the users trust each other and are willing to share the resources with each other. Each of these machines stores a large amount of data related to a domain and these resources are to be shared between a small numbers of trusted users. Also each user stores the resources in the file system in different directory structures, causing the same resource to be associated with different physical properties, thus making searching difficult.

Our work provides a solution to this problem so that users can browse through the resources stored by the users on their machines on the network by making use of what the resources talk about and not just by using the physical locations of the resources. Also the users can find the resources by giving such queries as "Get the documents that talk about anti-patterns and tactics written by John Doe and published by ABC between 2004 and 2006". This query will be executed on all the participating machines on the network and the resultant resources extracted by this query will be presented to the user. We also provide an easy to use graphical interface using movable objects representing queries, so that novice users can also make searches without the knowledge of querying.

Here, while talking about resources, we restrict ourselves to the management of documents and do not include other information objects such as emails, bookmarks etc., which can also be managed in a similar way.

Approach in Brief

The system takes as an input a Domain Knowledge Model (DKM) to create an interface for browsing through the resources. This DKM is a taxonomy in OWL format.

Based on this taxonomy the system implements a Semantic File System (SFS) which acts like a virtual file system. The SFS is a representation of the concepts of the domain in the form of a folder hierarchy. The resources are then mapped to this virtual file system based on the concepts that are described in the taxonomy or the DKM. To map the resources to the concepts they talk about, the system maintains some metadata describing the resources. This metadata contains the logical properties of the resources along with their physical properties.

Using these logical and physical properties of resources, the system generates a view to browse the resources, similar to that of windows explorer or a file browser, which is very much familiar to the users, reducing the learning curve.

Also these resources need to be searched for. So the system provides some easy to use search interfaces that the user can retrieve the resources based on both the physical and the logical properties of the resources.

Next, these resources and their metadata have to be made available to the other users in the network so that they can browse and search from these resources too.

The system is called *Semantic 'ls'* as it resembles the 'ls' command of UNIX in terms of functionality, with the options to the command being the domain specific and application specific meta data properties of the document rather than the physical properties of the file.

RELATED WORK

Several knowledge management systems for personal data have been developed by different people with varying functionality and features.

MyLifeBits [1] project aims to view personal data as a graph of information in which the nodes represent resources and metadata; and the edges represent the '*annotates*' relationship. Semantic 'ls' presents the resources as a directory structure where directories correspond to the concepts of the domain model and files correspond to the resources. MyLifeBits allows the users to annotate a file by linking it to another file that represents metadata and manually adding text or audio annotation

Haystack [2] is another Personal Information Management (PIM) system that supports annotations and collections. The concept of *collection* in this system is similar to that of *conceptualization of resources* in Semantic 'ls'. In the Haystack framework, the users can create, manipulate and visualize the metadata. Haystack creates the data model in a graph structure using the concept of semantic network, which essentially represents the metadata of the archived documents. The users can interact with this data model through a set of client services that visualize the metadata.

Among the existing approaches to integrate semantic web and PIM, the Gnowsis project [3] aims at semantic desktop environment that supports distributed data management using several Desktop Services. Gnowsis uses ontologies to express semantic associations and RDF for data modeling. However, the emphasis of Gnowsis is more on the flexible integration of a large number of applications into the semantic desktop environment than on semantic data organization and manipulation.

Semex [4] is another PIM System that uses resource annotation. Semex architecture has the features like auto extraction of the associations among resources, browse-by-associations and on-the-fly data integration. This architecture uses a single domain model of personal information, which has the concepts such as Person, Publication and associations such as Author-Of, while in Semantic 'ls' different application models are used based on the interest of the community.

SWAP [5] is a system that combines the semantic web and P2P technologies in knowledge management. Similar to Semantic 'ls', SWAP maintains a local repository at each node and the resources can be shared among peers. In this system, each node has a separate knowledge model and each query will be rewritten to fit various knowledge models. In Semantic 'ls', all peers follow a predefined knowledge model.

In [6] Vannevar Bush gives a general intuition of human thinking related to the Internet and the World Wide Web far before their actual invention.

SYSTEM ARCHITECTURE

The basic architecture of the proposed system (Figure 1) is 4-layered:

- The Data Layer – takes care of the file level needs within the system. Text-extraction from various types of files, metadata management and query handling are the major functions of this layer.
- The Network Layer – takes care of all the network related issues. Also it is used to provide an abstraction service to higher layers for maintaining transparency about the data location.
- The Semantic Layer – takes care of the semantic details of the resources. Construction of virtual directory structure based on the topics that each of the resources talks about and metadata maintenance for semantic querying are the major functions of this layer.
- The Visualization Layer – shows the concepts and the documents that talk about these concepts, as a directory structure, where in the directories correspond to the concepts and the files correspond to the conceptual documents.

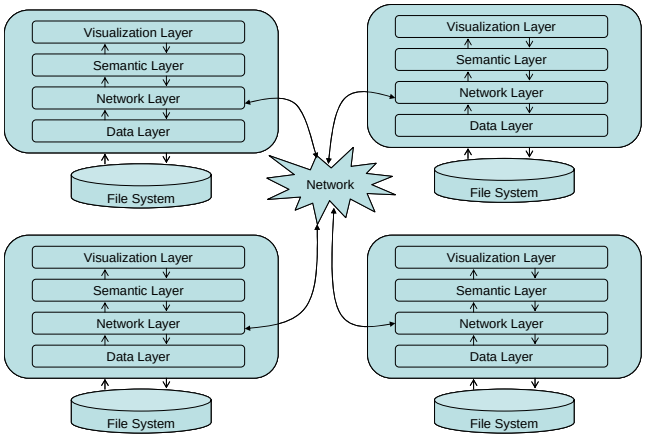

Figure 1 – Architecture of the proposed system in a P2P fashion

Data Layer

The data layer (Figure-2) implements 2 major functionalities: Query management on the metadata and file handling. To implement these functionalities, it makes use of: a Metadata management component which provides functions for operating on the metadata stored about the files, a query manager which makes use of the Metadata management component to convert the queries into a set of functions operating on the Metadata, a set of text extraction wrappers for various file types and a Document Loader component which is used to load the document as it is in the memory.

As part of query management, the Data Layer takes the input in the form of a query or an example object of the desired resource, which is converted into a list of basic functions provided by the metadata operations component.

As part of file handling, the data layer takes the physical path of the file as the input, which is got from the query management component and depending on the request the file is returned as an entire file by the *document loader* or extracted file by the *document wrappers*.

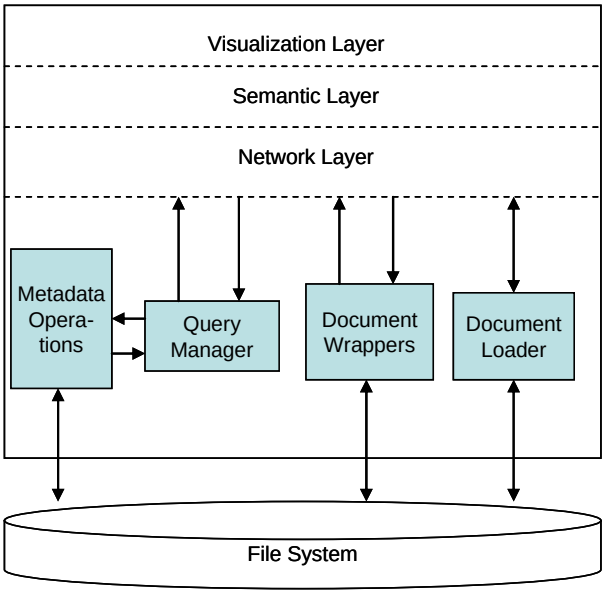

Figure 2 – Architecture of the Data Layer

Network Layer

Each personal digital repository contains a set of resources and it may not be feasible to integrate all the knowledge and store on all the systems. It would be advantageous to federate these systems whose users are interested in a common domain and form a community. Naturally, the systems in a community would be connected in a network using an intranet or the internet. So it is sufficient to form an overlay network on top of the existing network.

In the proposed architecture, the network layer (Figure 3) is responsible for communication with other nodes in the network. Each node provides a set of services to access its resources and annotation metadata so that other nodes are able to search or browse the shared resources.

All the nodes running *Semantic 'ls'* provide the same functionality and contain different sets of resources. Thus, each node can act as a peer in a typical peer-to-peer setup.

Discovery Service

Each peer has a discovery service using which the peers are able to create a list of currently active peers on the network.

Service for Remote Host

This component is used to get the metadata and the files stored on the remote machines. The Object Transfer Service is used to get the metadata and the File Transfer Service is used to get the entire files, both of which act as the semantic layer for the host on which the data resides and as a Data Layer for the host requesting the data.

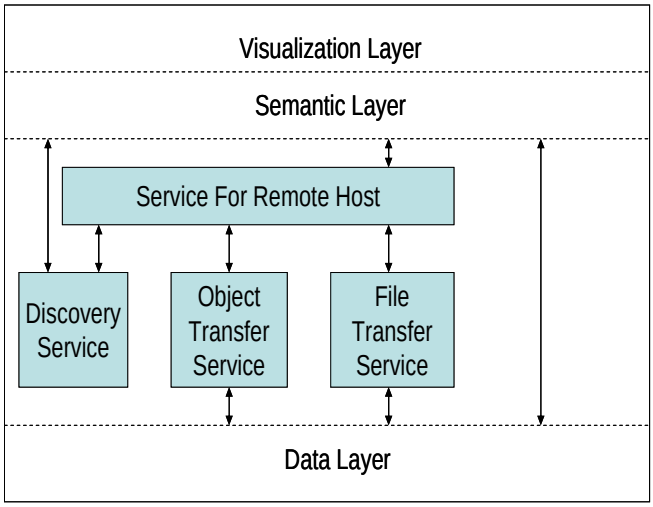

Figure 3 – Overview of the Network Layer

Semantic Layer

The semantic layer (Figure 4) has 6 components. The *SFS builder*, the *Resource annotator* and the *Query Translator* are the processes operating on the models called the *Domain Knowledge Model* (DKM) or the taxonomy, the *Annotation Schema* and the *SFS*.

The SFS Builder

The SFS builder works in two stages. First a sub-task called the *Structure Builder* within the SFS builder uses the concept hierarchy of the DKM, and builds the SFS. For this purpose, the *Structure Builder* uses the '*subclass-Of*' and '*super-Class-Of*' relationships of the DKM. Each concept in the DKM corresponds to a directory in the virtual directory structure. If a concept C1 is associated with another concept C2 with the subclass-Of relationship, then the directory D1 corresponding to C1 becomes a sub directory of the directory D2 that corresponds to C2. To process the DKM in OWL format, the system uses the Jena framework [7].

In the second stage, another sub-task of the SFS builder called the *Document Populator* uses the metadata of the documents returned by the data layer and populates the documents in the appropriate directories based on the *Concept-list* corresponding to each resource.

The SFS builder does not create any directories in the file system or change the physical location of the resources; it just helps the upper layers give a visualization of the directory structure to the user. Similarly the document populator too does not affect the physical properties of the file; it just helps in providing the higher layer with data for visualizing the resources in appropriate directories.

Resource Annotator

The utility of this component is not user-level. It just prepares the annotation metadata for all the resources. The query translator, to execute the semantic queries uses this metadata. A schema called the annotation schema or the application schema, which is generated through a user interface, governs the structure of the metadata.

The resource annotator works in two stages. First the form generator sub-task generates a form called the annotation form into which the user can enter values. And then the user input values are validated against the annotation schema before storing the input as metadata information for the resource.

The resource annotator also makes use of a simple automatic annotating algorithm that annotates the document based on the frequency of concepts that appear in the document and using the DKM as one of the references. This is a crude algorithm and sometimes still needs manual intervention, as not all the concepts of the resource are extracted, so we call it semi-manual annotation, as the user can change the concepts extracted using the form generated.

Query Translator

The query translator takes the user inputs and converts them into queries which are given to the bottom layers. The queries can include - extracting the appropriate metadata for a particular resource, extracting the text from the resource or opening a resource in its default application.

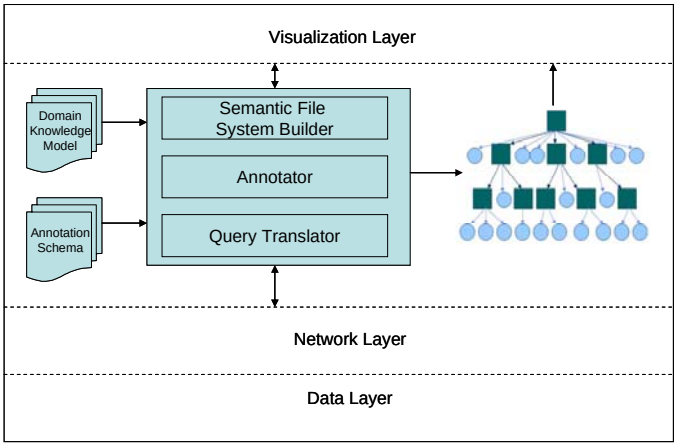

Figure 4 – Overview of the Semantic Layer

Visualization Layer

The visualization layer's function is to visualize the models of the semantic layer through some easy to use GUIs

(Figure 5). To realize this, a directory hierarchy is used to visualize the resources based on the annotation properties. We have used the "ArchVoc" taxonomy for Software Architecture from [8] for creating the DKM.

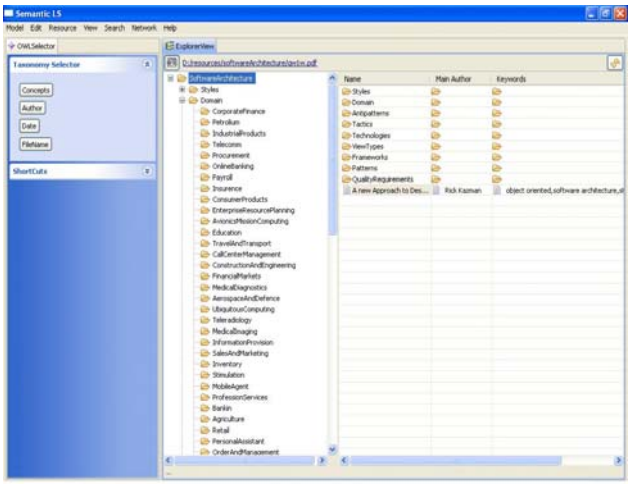

Figure 5 – Interface for browsing the resources

The layer also provides views for searching a particular resource. There are 2 types of search – a QBE search view and a search view based on Boolean Logic. In the QBE search view the user is able to make advanced queries to search for the resources by specifying the properties of the resource. The Boolean logic based search view (Figure 6) is an advanced version of [9]. Here the query is formed by creating a visual object for each property value and selecting the required areas which are to be searched. The visual objects can be dragged around thus making them intersect with each other and providing more areas to select. To keep the complexity/cognitive load low, we have kept the maximum visual objects in a query to be four.

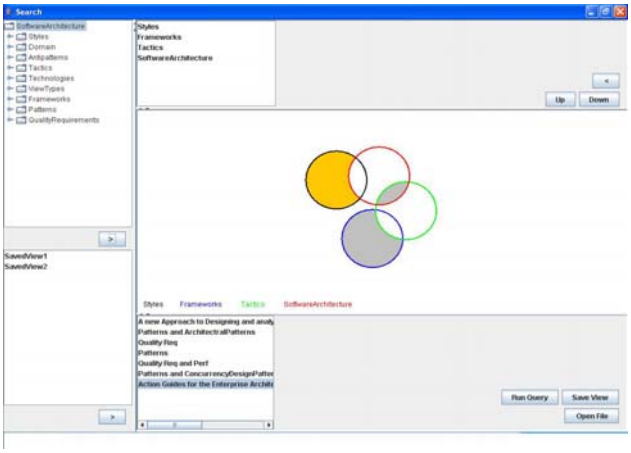

Figure 6 – Interface for Boolean search

If the number of parameters in the Boolean Search is to be greater than four, then a facility for saving a partial query of up to four visual objects as a new visual object is given, so that it can be again used to form a new query where the number of objects are more than four.

Another view given to the user is the annotation view where the user can annotate the resources by adding values to the properties of the resource. This is the view used to specify the metadata for each resource.

IMPLEMENTATION

The entire system is written in Java. The technologies used for writing the system are Eclipse, DB4O, Jena, and Swallow.

All the metadata management functionality uses DB4O [10], an Object Oriented Database for serialization of the objects, for storing metadata information and querying on the metadata. The semantic layer is mostly coded in pure java along with some functionality provided by Jena [7] for handling the OWL files or the DKM data files for creating the virtual file system. This entire system is deployed as an Eclipse product; so the visualization layer makes use of the Eclipse RCP [11] framework to provide a very rich user interface. The network layer has been implemented by making use of Swallow [12] which provides a container where P2P applications can be deployed and a discovery service called Federated Advanced Directory Architecture (FADA) [13] that keeps the list of peers running a particular service, but we are changing the network layer to work with similar functionalities but using a much lighter framework for P2P communication.

CONCLUSION AND FUTURE WORK

A personal information management system has been proposed with a set of semantic based features, over a P2P network extending it to a private group information manager. Our motivation in architecting such a system is to define a way to provide an information management platform for groups working on similar domain, who are not necessarily advanced users or trained users. The issues involved in realizing the system have been studied. Easy to use and familiar user interfaces have been designed to reduce the learning effort for the users. Efforts have been made to make searches more than just keyword based and easier to formulate. We are also developing interfaces and techniques for creating and maintaining the taxonomy.

Our work gives a framework of how semantic techniques can be used for information management at both, a single user level, as well as a multi user level and the work can be extended in many areas. The current architecture of the system considers only a single P2P network. In future it can be extended to multiple P2P networks. At present the system deals only with the *subclass-Of* and *super-Class-Of* relationships of the domain model. But the domain knowledge model could contain other relationships as well. To deal with these relationships, a new UI paradigm has to

be thought of. The metadata required for the semantic search is now prepared through semi-manual resource-annotation. Automating the process of resource annotation could be a major improvement. This implementation manages resources that are in the form of documents, but it can be extended, using similar techniques, so that other personal information such as emails, bookmarks, chat sessions etc. can also be handled.

ACKNOWLEDGEMENTS

This work has been funded by The European Union's 6th Framework Program of research under the project *O*pen *P*hilosophies for *A*ssociative *A*utopoietic digita*L* ecosystem*S* (OPAALS).

REFERENCES

1. Jim Gemmell, Gordon Bell, Roger Lueder, Steven M. Drucker, and Curtis Wong. MyLifeBits: fulfilling the Memex vision. In ACM Multimedia, pages 235–238, 2002.
2. Dennis Quan, David Huynh, and David R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In Proceedings of ISWC, 2003.
3. Leo Sauermann. The Gnowsis-Using Semantic Web Technologies to build a Semantic Desktop. Diploma thesis, Technical University of Vienna, 2003.
4. Xin Dong and Alon Y. Halevy. A Platform for Personal Information Management and Integration. In CIDR, pages 119–130, 2005.
5. M. Ehrig, C. Tempich, J. Broekstra, F. van Harmelen, M. Sabou, R. Siebes, S. Staab, and H. Stuckenschmidt. SWAP - ontology-based knowledge management withpeer-to-peer technology, Proceedings of the 1st National Workshop Ontologie-basiertes Wissensmanagement (WOW2003).
6. Vannevar Bush. As We May Think. The Atlantic Monthly, volume , July 1945.
7. Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/
8. Lenin Babu T, Seetha Ramaiah M, Prabhakar T.V., Rambabu D. ArchVoc - Towards an Ontology for Software Architecture. International Conference on Software Engineering; Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, 2007.
9. Boolistic , http://www.boolistic.com
10. DB4O , http://www.db4o.com/
11. Eclipse RCP, http://www.eclipse.org/rcp/
12. Servent , http://swallow.sourceforge.net/.
13. Federated Advanced Directory Architecture (FADA), http://fada.sourceforge.net/.