A Multi-Ontology Approach for Personal Information Management *

Huiyong Xiao Isabel F. Cruz

Department of Computer Science
University of Illinois at Chicago
{hxiao | ifc}@cs.uic.edu

Abstract. The increasingly huge volume of personal information stored in a desktop computer is characterized by disparate models, unstructured contents, and implicit knowledge. Aiming at a semantic rich environment, a number of *Semantic Desktop* frameworks have been proposed, concentrating on different aspects, including organization, manipulation, and visualization of the data. In this paper, we propose a layered and semantic ontology-based framework for personal information management, and we discuss its annotations, associations, and navigation. We also discuss query processing in two cases: query rewriting in a single personal information application, PIA, and that between two PIAs.

1 Introduction

In 1945, Vannevar Bush put forward the first vision of personal information management (PIM) system, Memex, by pointing out that the human mind "operates by associations", and we should "learn from it" in building Memex [4]. The Hypertext systems (see the survey of Conklin [6]), which flourished in the 80's, reinforced this vision and yielded the current World Wide Web, in a broader scope. Recently, with the Semantic Web vision [2], a number of PIM systems associated with that vision, hence called *Semantic Desktop*, have been proposed. By summarizing these proposals and taking into account the characteristics of personal information (PI), we propose the following principles that a PIM system should follow:

Semantic data organization. Almost all existing approaches are trying to go beyond the hierarchical directory model. The critical factors of semantic data organization include adequate annotations, explicit semantics, meaningful associations, and a uniform representation. A semantic-rich data organization has several advantages. First, the annotations and associations (as the *superimposed* information over the coarse data [19]) form the context of the PI, thus making the data more easily understandable. Second, the superimposed information also allows for a finer and more flexible manipulation (e.g., browsing and querying) of the data. Third, an explicit formal semantics for the data can facilitate reasoning on the data and deriving new knowledge. Finally, the uniform representation can support the integration of data that may be heterogeneous.

Flexible data manipulation. A PIM system can provide integration, exchange, navigation, and query processing of the stored personal information. The framework of PIM,

* This work was partially supported by NSF Awards ITR IIS-0326284 and IIS-0513553.

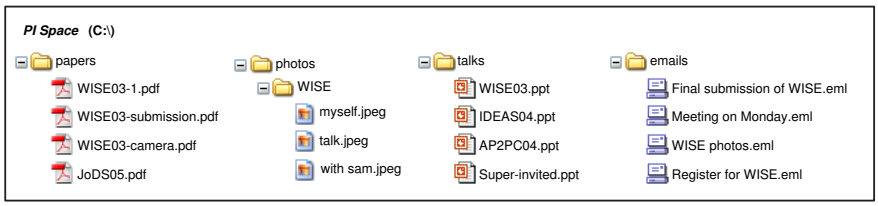

Fig. 1. An example of files in a PI space.

including the data model, query language, and user interface, should provide multiple ways to manipulate data in a powerful and flexible manner. Furthermore, a PIM system should possess the capability for seamless communication (or interoperability) with external sources (possibly in another PIM system), e.g., in a peer-to-peer (P2P) way [26].
Rich visualization. Multiple visualizations can help the user in understanding data. Instead of providing separate views of the data as most traditional applications do, a PIM system should support data visualization from different perspectives, to offer a comprehensive view. Examples include association-centric visualization [24] and time-centric visualization [13, 12].

Example 1. Figure 1 presents a fragment of PI space, which consists of four directories of files in the hard drive C:\. The papers directory contains four papers of the format pdf, photos\WISE contains three pictures taken at the WISE '03 conference, talks contains four Powerpoint files that are respectively the slides of four talks, and emails contains four saved email messages. Even if the concrete contents of all these files are unknown, we can tell from their names (or the names of their respective directories) that several of them appear to be related to one another. Unfortunately, their storage in different and possibly unrelated directories does not show such inter-relationships, thus resulting in possible difficulties in locating the wanted information. Some keyword-based searching techniques, e.g., offered by the Google Desktop Search,[1] can retrieve all files that are relevant to WISE. However, without further inspection of the contents of each file, the user may not be able to discover certain associations between them, e.g., that file JoDS05.pdf is an extended journal paper of WISE03-camera.pdf.

From this example, we can see that the lack of semantic associations among the stored data could be a handicap for data and knowledge discovery. In this paper, we focus on issues of semantic data organization and management in PIM, by taking the following approach:

1) We propose a layered framework for PIM, in which multiple ontologies playing a variety of roles are employed. Specifically, the *resource layer* stores all the PI resources (using URIs), metadata of the PI, and all kinds of associations using RDF. The *domain layer* contains the ontologies specific to various domains that are used to structure the data and categorize the resources. The *application layer*, built on top of the domain layer, is where the user constructs different application ontologies for different purposes of data usage. This layered architecture enables: i) a semantics-rich environment for

[1] http://desktop.google.com

personal information management; ii) a flexible and reusable system, by decoupling the domain and application ontologies, so that the construction of application ontologies for different applications can reuse the underlying domain ontologies. We argue that this provides certain advantages over the use of a single domain model for all the PI (e.g., [9]).

2) We discuss in detail how to utilize superimposed information for semantic organization, focusing on the construction of resource-file and resource-resource associations. We also present the idea of *3D navigation*, which is a combination of the *vertical*, *horizontal* and *temporal* navigation in the PI space. The idea is inspired by some existing PIM systems including MyLifeBits [13] and Placeless Documents [10], and is demonstrated in a browser.

3) In our framework, the basic unit for the user to manage the Semantic Desktop is the *personal information application* (PIA). Each PIA aims to accomplish or assist a specific task (e.g., bibliography management, paper composition, and trip planning). The PIAs can be standalone, with their own application ontology, user interface, and workflows. Meanwhile, they can communicate with each other as if in a P2P network, by means of the connections (mappings) established between their application ontologies. In this sense, different PIAs interoperate at a semantic level. We describe query processing in our framework in two cases: within a single PIA or between two PIAs, in a P2P query processing mode.

Among the existing approaches to PIM, the Gnowsis project from DFKI[2] aims at a Semantic Desktop environment, which supports P2P (or distributed) data management based on *Desktop services* [26]. Like in our framework, Gnowsis uses ontologies for expressing semantic associations and RDF for data modeling. However, the emphasis of Gnowsis is more on the flexible integration of a large number of applications than on semantic data organization and manipulation. SEMEX [9] is another personal data integration framework that uses data annotation (i.e., schemas), similarly to our ontology-based framework. A single domain model is provided as the unified interface for data access.

MyLifeBits [13], Haystack [24], and Placeless Documents [10] are three PIM systems that support annotations and collections. Here, the concept of *collection* is essentially the same as the conceptualization (using ontologies) of resources in our framework. MyLifeBits supports easy annotation and multiple visualizations (e.g., detail, thumbnail, timeline, and cluster-time views on the data). For this purpose, the resources are enriched by a number of properties, including the standard ones (e.g., size and creation date) and more specific ones (e.g., time interval) [13]. Haystack aims to create, manipulate, and visualize arbitrary RDF data, in a comprehensive platform. For visualization, it uses an ontological/agent approach, where user interfaces and views are constructed by agents using predefined ontologies [24]. Placeless Documents introduces "active properties", where documents can have executable codes that provide document-based services [10].

Other approaches to PIM include Chandler,[3] Lifestreams [12], Stuff I've Seen (SIS) [11], and Xanadu [22], which focus on different aspects.

[2] http://www.dfki.de/web/

[3] http://www.osafoundation.org/

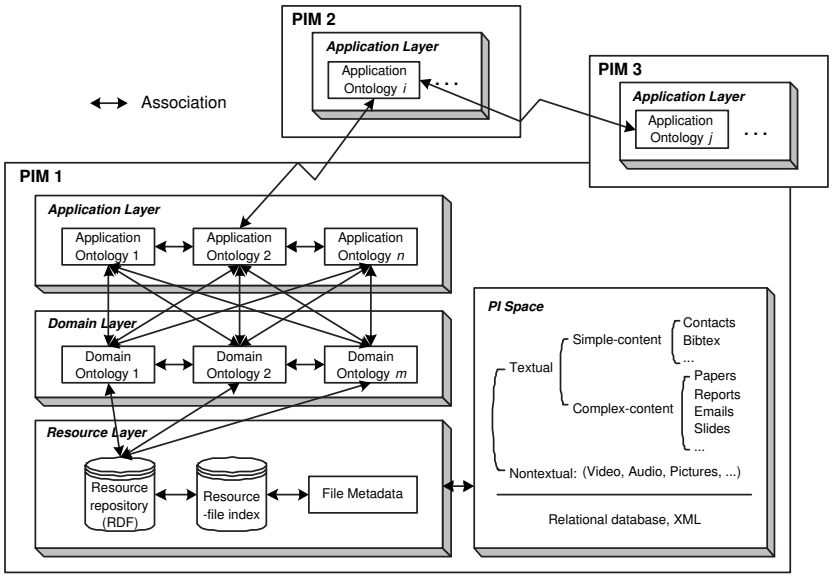

Fig. 2. An ontology-based framework of a PIM system.

The rest of the paper is structured as follows. In Section 2, we describe the layered framework and its main components. The semantic organization of the PI (including the concepts of annotation, association, and representation) is discussed in Section 3. Section 4 and Section 5 focus on two main ways of data manipulation, namely, navigation and query processing. Finally, we conclude in Section 6.

2 Framework

Our framework follows the principle of superimposed information, i.e., data or metadata "placed over" existing information sources [19]. This concept seems particularly useful for the organization, access, interconnection, and reuse of the information elements. We propose for PIM a layered ontology-based framework, as shown in Figure 2, with the following data components:

Personal information space. The personal information space may contain structured data (e.g., relational), semi-structured data (e.g., XML), or unstructured data. Unstructured data can be textual or non-textual (as in video, audio, or picture files). Furthermore, textual files can be classified as simple-content or complex-content. More specifically, simple-content files have no references to other files. Typical examples include people contacts and Bibtex entries. In contrast, complex-content files have a flexible scheme of presentation, and may contain references to other files, e.g., by means of citations or hypertext links [6]. For example, a paper in the PI space may cite another paper (existing in the PI space or an external space), which, in turn, could cite other papers.

File description. We annotate each file using a file description (or metadata) consisting of a set of properties of the file. Each item in the file description is a property-value pair. The file description is the first-level (direct) annotation for the individual files, and has the same scheme (structure) for the same type of files. For example, the following fragment contains a typical description of a JPEG file.

```
Dimensions:   3072 × 2048 pixels
Device make:  Canon
Color space:  RGB
Focal Length: 75
......
```

Domain ontologies. A number of ontologies are published on the Web. Examples of such ontology libraries include DAML Ontology Library,[4] the Semantic Web Ontologies,[5] and the Protégé OWL ontologies.[6] The ontologies in these libraries are typically designed and organized for different domains such as Conference, Person, Photo, and Email. In our framework, the domain ontology layer is designed to be loosely-coupled with the other layers, to enable the insertion and removal of ontologies as "plug-ins".

Resource-file index and RDF repository. One of the roles of domain ontologies is to provide the basis for data classification. In order to establish the connections between the files and the concepts in the domain ontologies, we treat each file as a resource, which is then classified as an instance of one or more concepts. The resource-file index is a local database storing these connections between resources and files. Furthermore, the various types of associations among resources (as instances of *association of concepts* in the domain ontologies) are stored in an RDF repository. The resource-file index and the RDF repository are both in the *resource layer*, providing resource instances for the domain ontologies in the *domain layer* above.

Application ontology. Above the *domain layer* is the *application layer*, which contains the ontologies for different applications. The domain ontologies, as an intermediate layer between the applications and the data, are meant to enhance the reusability and flexibility of the framework. More specifically, the application ontologies are defined as views of the domain ontologies, which can be reused for the construction of different application ontologies. In our framework, each *personal information application* (PIA), is associated with an application ontology, has access to relevant data, and is functionally independent of other applications. It may be infeasible to have a single ontology to cover various applications, e.g., for trip planning and paper writing. Instead, as many PIAs as needed can be designed in one or more PIM systems, where the PIAs can interoperate (e.g., through P2P query processing) for the purpose of integrating relevant information. This issue is elaborated on in Section 5.

Besides the data components described above, a PIM system also needs some functional components to perform all kinds of data and metadata processing, to make the framework work as a whole. Such components include an *indexer* (for establishing and managing the indexes of the files), a *wrapper* (for identifying and extracting resources

[4] http://www.daml.org/ontologies/

[5] http://www.schemaweb.info

[6] http://protege.stanford.edu/plugins/owl/owl-library/

from the files), and an *ontology designer* (for importing and editing an ontology). Because of space limitations, we do not elaborate further on these components.

3 Semantic Data Organization

The layered architecture of our PIM framework described previously enables the reusability and the organization of semantically rich data for PIM. In this section, we discuss in detail the mechanisms that our framework uses to support the semantic organization of the PI space, including those for semantic annotation, association, and representation.

3.1 Annotation

Given that the data in the PI space is the base information, all the other data components in our framework are actually superimposed information over this base. The most fundamental function of the superimposed information is to provide semantic annotations of the base information to enable powerful and accurate data access. We discuss the following two aspects:

File description. It is especially important to provide the searcher with a detailed description of the nontextual files. When performing a keyword-based searching, the searcher matches the submitted keywords (e.g., "Canon") or key-value pairs (e.g., "Maker:Canon") with the property-value pairs of the file description, to find the right files requested by the user. Even for textual files, taking into account such metadata will improve the effectiveness of full-text searching.

Domain ontologies. Given that a file is identified as a resource, we are able to annotate the file using a domain ontology, by associating the resource with a concept of an ontology. The domain ontology provides not only a context for understanding the data, but also semantic clues for the precise data retrieval. For example, the user can query the PI using a query language for RDF instead of using keywords. We note that a file can be an instance of more than one concept, according to different classification criteria.

3.2 Association

In our framework, semantic associations are used to relate all the data (base information) and metadata (superimposed information). There are two classes of associations: the *resource-file associations* that are actually the resource-file indexes and the *resource-resource associations* that are instances of the domain ontologies and are stored in the RDF repository.

Resource-file associations. In addition to the ontological resources that are used to identify (through data classification) the files, a (textual) file may contain and refer to a number of resources. Therefore, the resource-file associations can be one of the following: *identification*, *containment*, and *reference*.

Example 2. Suppose that the user has saved an email message, which is an announcement of a seminar, as shown in Figure 3. First, the email message can be classified as an instance of the concept Email, provided that the concept exists in some domain ontology. Then, the system can generate for the concept SeminarAnnouncement and its

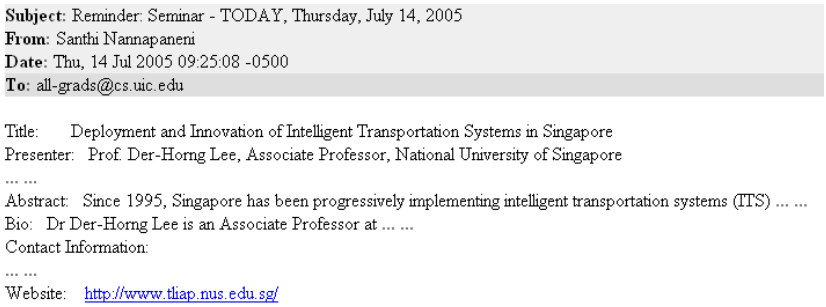

Fig. 3. An example of an email message.

properties a new instance (i.e., resource), which is associated with the saved email by the relationship *containment*. Finally, a *reference* association can be established between the resource http://www.tliap.nus.edu.sg/ (e.g., of the concept WebsiteAddress) and the email message.

The process of setting up the resource-file associations is the one of recognizing resources from the file description and/or the file content and then mapping them to the ontological concepts. The user may determine the degree to which the resources should be extracted from a file and its description. For instance, in the previous example, the user can further create resources for the title and abstract of the seminar, and for the biography of the presenter. It is expected that this process (as well as the process of discovering resource-resource associations, as discussed later) can be maximally automated, to reduce the user's burden. For this purpose, we may utilize the following methods:

- **Keyword extraction**. From the text of a file, keywords can be extracted based on a thesaurus or be highlighted manually by the user. Each keyword can be considered a resource contained by the file. The matching of the resources with the concepts in the domain ontologies can be guided by a thesaurus such as WordNet.[7]
- **Hyperlink analysis**. For the textual files that include hyperlinks to classified resources (e.g., a citation of a paper or a link to a webpage), we create for each hyperlink a reference-type resource-file association, as well as a resource-resource association between the referring resource and the referred one.
- **Natural language processing**. We can utilize known techniques (e.g., [1]) to parse each sentence of a text or its summary obtained by means of text summarization [20]. For each resulting triple $\langle subject, predicate, object\rangle$, we try to match it with the patterns $\langle s, p, o\rangle$ in the domain ontologies, where p is a property of the concept s and has a value typed of o. If such pattern exists, a resource-resource association of type *property* and of the form $\langle subject, predicate, object\rangle$ is generated.
- **History**. As the framework proceeds with such classification and cognition, more and more knowledge about this process can be accumulated and reused by a new process.

[7] http://wordnet.princeton.edu

Table 1. Resource-resource associations.

Resource-resource associations	Intra-domain	Inter-domain	Intra-application	Inter-application	Domain-application
aggregation	√	√	√		
property	√	√	√		
instantiation	√	√	√		
generalization	√	√	√		
ontology mapping		√		√	√

Resource-resource associations. We borrow from the Object Oriented Design (OOD) techniques the following four types of relationships between objects: *instantiation* (i.e., membership), *property*, *aggregation* (i.e., whole/part), and *generalization* (i.e., inheritance). These four relationships, which are used in object models, are adopted to describe the associations among concepts as well as resources in our framework. Note that "property" refers to a pattern as identified, for example, using natural language processing techniques, which corresponds to a user-defined property. For example, writes can be a property of the concept Author, connecting Author to the concept Book. Table 1 summarizes the resource-resource associations in our framework.

By using the previously described techniques, we can discover the resources and their associations implied in the PI, and classify them into the domain ontologies, thus populating the ontologies. In the example of Figure 3, it is possible to extract a pattern ⟨Singapore, implements, ITS⟩, which could then be classified as an instance of an ontological pattern such as ⟨Organization, implements, System⟩, where Organization and System are two concepts, and implements is a property. Note that the user is allowed to choose the granularity of this knowledge (resource and associations) discovery process, ranging from only taking the whole file as a single resource to analyzing the detailed contents of the file.

In addition, ontology mappings may be established between correspondences that connect concepts in different domain and application ontologies. Currently, we consider *equivalence* as the only semantics for the mapping between two concepts, although richer semantics of the mappings could be considered [17].

3.3 Representation

In our framework, all information, including file descriptions, the resources in the repository, and the resource-file indexes, are represented in the Resource Description Framework (RDF),[8] a W3C proposed standard. For the schema of these data (i.e., the application and domain ontologies), we use the vocabulary language for RDF, RDF Schema (RDFS).[9] The RDF model is a semantic network, where the nodes denote the resources and the edges are properties that represent the relations between resources. The network can also be seen as a set of statements (triples) in the form of (subject, predicate, object). RDFS is used to define the vocabulary (in terms of classes and properties) of

[8] http://www.w3.org/RDF/

[9] http://www.w3.org/TR/rdf-schema

Table 2. RDF properties for the associations.

Relationship	RDF property	Comments
aggregation	rdfx:contains	rdfx is the abbreviation of the namespace, where the property contains is defined. For example, <#a, rdfx:contains, #b> means that a contains b.
property	User-defined properties	For example, <#wise03talk, presentedBy, #xiao> means that wise03talk is connected to xiao by the association presentedBy.
instantiation	rdf:type	For example, <#xiao, rdf:type, #Person> means that the resource xiao is an instance of the concept Person.
generalization	rdfs:subClassOf	rdfs:subPropertyOf is used for property generalization.

the RDF data, such as rdfs:Class, rdf:Property, and rdf:type. Table 2 summarizes the RDFS vocabularies that are used to represent different types of associations.

The use of RDF as the data model and RDFS as the ontology language in our framework is motivated by the nature of the RDF as a Web resources description mechanism and the fact that the PI is represented as a set of interrelated resources. In contrast, XML is not chosen because it cannot represent semantic associations [8]. Certainly, OWL (Web Ontology Language), as built on top of RDFS, is more expressive for ontology representation. However, the use of a slightly extended version of RDFS is adequate for representing resource-file and resource-resource associations.

The extension to RDFS is as follows: we define in a namespace (abbreviated using the prefix rdfx) a new RDF property, contains, which is used to represent the *aggregation* relationship. For the representation of the *instantiation* and *generalization* relationships, we use rdf:type and rdfs:subClassOf, respectively. The *property* relationship is represented naturally by an RDF property defined in the user-defined namespace. Figure 4 gives a concrete example of an RDF representation.

4 Semantic Navigation

It is critical for a Semantic Desktop to provide the user with the capability to access the stored data in a variety of ways. The user may want to browse the information by means of the flexible and intelligent navigation in the information space, including the base and superimposed information. The user may also desire that certain query facilities (e.g., keyword-based searching or certain query languages) be provided by the framework. In this section, we discuss the navigation in the data space of a Semantic Desktop. Query processing is discussed in the next section.

The semantic data organization in our framework enables the navigation in the PI space, making use of useful hints (e.g., the context of a concept being browsed) so as to facilitate the user's understanding of data. More specifically, by taking into account the layered architecture, the semantic navigation in our framework can be performed in three directions: (1) In *vertical navigation*, the user follows a path across layers. Two cases are possible for this way of navigation: top-down from the application ontologies to the stored files and bottom-up from the stored files to the application ontologies. (2)

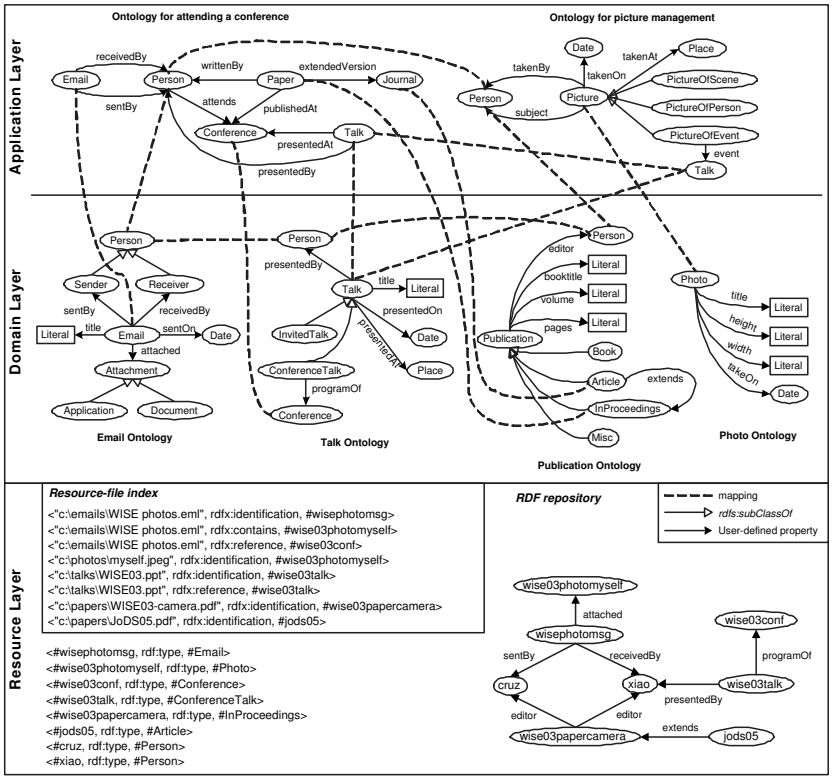

Fig. 4. Representation of the application, domain, and resource layers. All ontologies are represented in RDFS. Two application ontologies for PIAs, i.e., picture management and publication management, are constructed. Below them are four ontologies for the domains of Email, Talk, Publication, and Photo, respectively. At the bottom, the resource-file and resource-resource associations are represented as triples or in a graph.

In *horizontal navigation*, the user follows links of concepts (or resources) within one layer. Typically, there are three cases of horizontal navigation, corresponding to each layer: application-to-application navigation, domain-to-domain navigation, and file-to-file navigation. (3) In *temporal navigation*, the user can navigate by following references in chronological order, each being a resource for the same real world object with a time stamp associated with it. For example, the user may want to look at different versions of a research paper.

All the base and superimposed information in the framework forms a directed graph, where the vertices are the resources in the ontologies and the files stored in the PI space, and the edges are the associations between the resources and files. We say that the three directions of navigation together provide the capacity of a *3-dimension* (3D) navigation mechanism, which can facilitate the construction of a browser. For instance, suppose the user is browsing a specific application ontology in a visualized browser. When the user clicks on the node of a concept in the ontology, the browser can then choose to

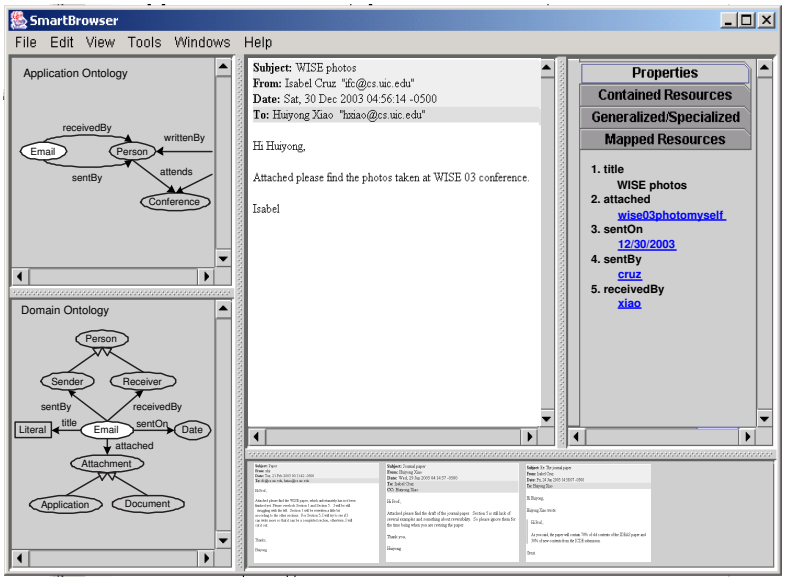

Fig. 5. The browser for PIM.

display the instances of the concept thus selected (by vertical navigation), the context of the concept in the domain (also by vertical navigation), and the associated concepts in other application ontologies (by horizontal navigation). Compared to the traditional navigation approach that is based on hierarchical directories, 3D navigation is based on semantic associations, similarly to those that humans establish between concepts.

Example 3. Consider the scenario shown in Figure 4. The spirit of 3D navigation is demonstrated in the browser of Figure 5. The *current* resource (file) that the user is browsing is an email message (i.e., wisephotomsg), which has some photos attached, which were taken at WISE '03. The concepts that this resource belong to are highlighted (in white) so as to show the contexts to which they belong. All associated resources are categorized and shown on the right tabbed pane, which provides a guidance for the user in navigating the PI space. The bottom-right pane shows the timeline of different versions of the current resource (if they exist) or all the resources belonging to the same concept as the current resource.

5 Semantic Query Processing

Unlike navigation, which is an interactive process, query processing is performed without further intervention from the user. To retrieve relevant data from the PI space, the user's request may be posed as a sequence of keywords or as a query formulated in a certain query language.

The keyword-based search matches the input keywords and the vector of words in the *candidate* documents, calculates the *similarity* for each of the matches, and returns

to the user the results after *ranking* them [25]. The results of a search are usually evaluated using the statistical criteria such as *precision*, *recall*, or a combination of them. The shortcoming of keyword-based search is that the semantic associations between relevant data are not considered. In contrast, query languages can provide a semantically richer access interface, thus facilitating the data retrieval and improving the accuracy of the answers. However, a query is usually performed based on an exact match between the query and the data, so that the recall of the answers is influenced, in the sense that some relevant but not matched data is not retrieved.

Since the two approaches complement each other, it is desirable to provide both of them. In this section, however, we mainly focus on query processing in our framework. We choose to express the queries in RDQL [16]; they can query both the resources and their associations. We discuss how to process a query submitted by the user in two cases: within a PIA and across different PIAs.

5.1 Query processing in a PIA

In our framework, the user query is formulated in RDQL (RDF Data Query Language), which uses an SQL-like syntax [16]. To reduce the user's burden, a graphic means can be used to facilitate the user's query formulation. For simplicity, we use a subset of RDQL that we call *conjunctive RDQL* (c-RDQL), which can be expressed as a conjunctive formula: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n})$, where $\boldsymbol{X_i} = (x_i, x_i')$ and p_i is an RDF property of x_i having the value x_i'.

In our framework, an application ontology is constructed over one or more domain ontologies, and the files in the PI space are formalized as instances of the concepts in the domain ontologies. If we consider the application ontology as the global ontology (since the user query is posed on it), the whole system can be seen as a GaV data integration system [18]. Therefore query processing in a single PIA is performed as in a GaV system. In particular, when the user poses a query (in RDQL) over the application ontology, the RDQL query is then rewritten into a new RDQL query in terms of the domain ontologies, based on the mappings between the global ontology and domain ontologies. By executing the rewritten query on the corresponding domain ontologies, resources (files) that match the query are then returned as answers to the query.

There are a number of algorithms for query rewriting in relational or XML data integration systems [14]. In a GaV based integration system, query processing is performed using a "unfolding" strategy [18]. More specifically, for rewriting a query (e.g., a conjunctive query) that is posed on the global schema or ontology, we simply substitute the predicates in the body of the query with the corresponding view definitions. In our framework, where the mappings between the application ontology and the domain ontologies are expressed as RDF class or property correspondences, the algorithm for query rewriting is similar to this strategy.

By assuming that there are no integrity constraints over the application ontologies and the user queries are formulated in c-RDQL, we give the formal description of our query rewriting algorithm in a single PIA, which we call ADREWRITING (for rewriting from Application ontologies to Domain ontologies), as follows. We note that we do not consider the namespaces of ontologies for simplicity of the description.

Algorithm ADREWRITING

Input:1. q_1 over the application ontology $\mathcal{G}$: $ans(\boldsymbol{X})$:- $p_1(\boldsymbol{X_1}), ..., p_m(\boldsymbol{X_m})$;
2. $\mathcal{M}$: the mapping table between $\mathcal{G}$ and domain ontologies $\mathcal{S}_1, ..., \mathcal{S}_n$.

Output: q_2: A c-RDQL query over $\mathcal{S}_1, ..., \mathcal{S}_n$.

1. $head_{q_2} = ans(\boldsymbol{X})$; $body_{q_2} = null$;
2. **For** $i = 1$ **to** m **do**
3. (c_1, c_2) = name of the classes referred to by (x_1, x_2), for $\boldsymbol{X_j} = (x_1, x_2)$;
4. Search $\mathcal{M}$ to find (d_1, d_2) such that $\{(c_1, d_1), (c_2, d_2)\}$ are two class correspondences in $\mathcal{M}$;
5. Traverse $\mathcal{S}_1, ...,$ and $\mathcal{S}_n$ by following all kinds of associations, to find the vertices, $v_1, ..., v_k$, connecting from d_1 to d_2;
6. **If** $k = 0$ **then** add $p(x_1, x_2)$ (or $p(x_2, x_1)$) to $body_{q_2}$, if there exists p connecting d_1 to d_2 (or d_2 to d_1);
7. **Else for** $j = 1$ **to** $k - 1$ **do**
8. Add $p(\hat{x}_j, \hat{x}_{j+1})$ (or $p(\hat{x}_j + 1, \hat{x}_j)$) to $body_{q_2}$, if p is not a mapping and connects v_j to v_{j+1} (or v_{j+1} to v_j);
9. Add $p(x_1, \hat{x}_1)$ (or $p(\hat{x}_1, x_1)$) to $body_{q_2}$, if p is not a mapping and connects d_1 to v_1 (or v_1 to d_1);
10. Add $p(\hat{x}_k, x_2)$ (or $p(x_2, \hat{x}_k)$) to $body_{q_2}$, if p is not a mapping and connects v_k to d_2 (or d_2 to v_k);
11. $q_2 = head_{q_2}$:- $body_{q_2}$;

Example 4. Suppose the user wants to list all conference papers with their authors and journal version, using the query $q_1 : ans(x, y, z)$:- $writtenBy(x, y), extendedVersion(x, z)$, which is posed on the application ontology of publication management. For the variables (x, y, z), we get the classes that they refer to as (Paper, Person, Journal), as indicated by Line 3. By looking into $\mathcal{M}$, we find the corresponding class sequence as (Publication:InProceedings, Publication:Person, Publication:Article), where the names before the colons are domain ontology names. From Lines 5 to 10, we compute the predicates in the body of q_2 as follows.

q_2: $ans(x, y, z)$:- $editor(x, y), extends(z, x)$

By executing q_2 over the RDF repository as shown in Figure 4, we get the answer {(#wise03papercamera, #xiao, #jods05), (#wise03papercamera, #cruz, #jods05)}.

5.2 A2A query processing

Application to application (A2A) query processing occurs when an application is attempting to retrieve relevant data from another semantically related application, to answer a query. If the PIAs are considered as connected peers (i.e., service providers for certain data access), the A2A query processing is similar to that in peer-to-peer (P2P) systems [7, 15]. Whether the PIAs exist in a single desktop or are physically distributed makes no differences to the A2A query processing.

A2A query processing consists of two steps of query rewriting. First, we rewrite the original query q, which is posed on the application ontology $\mathcal{G}_1$, to a query q' on the other application ontology $\mathcal{G}_2$, according to the mappings between $\mathcal{G}_1$ and $\mathcal{G}_2$. Then, q' is rewritten to a query q'' on the domain ontologies, to which $\mathcal{G}_2$ is mapped. Answers are obtained by executing q'' on the RDF repository. The second query rewriting is exactly the one described by the algorithm ADREWRITING, whereas the first rewriting is slightly different from ADREWRITING. In particular, unlike the total mapping from an application ontology to the domain ontologies, some of the concepts in $\mathcal{G}_1$ may not be mapped to those in $\mathcal{G}_2$. Therefore, the answers returned by q'' may contain null values or Skolem functions for the unmapped concepts or properties.

The A2A mappings can be derived by composing the mappings between $\mathcal{G}_1$ and the domain ontologies, inter-domain mappings, and those between $\mathcal{G}_2$ and the domain ontologies. To evaluate both query rewriting processes, we need to check the equivalence (or containment) between a query and its rewriting. A *correct* query rewriting is the one that is equivalent to (or maximally contained in) the query. These two issues (*reasoning on mappings* [3, 23] and *reasoning on queries* [5, 21]) have been extensively studied and are beyond the scope of this paper.

6 Conclusions and Future Work

In this paper, we present our design of a PIM system. We propose a layered ontology-based framework, which aims to provide a semantics-rich environment for personal information organization and manipulation. The multiple ontologies existing in different layers of the architecture explicitly support the data semantics. Furthermore, the decoupling of the domain layer and the application layer enhances the flexibility and reusability of the framework. Specifically, we discuss in detail the semantic-enriched data organization, including the use of file descriptions and domain ontologies as annotations, and the construction of resource-file and resource-resource associations. We also introduce the idea of *3D navigation*, which is used in a desktop browser. We discuss query processing in our framework in two cases: within a single personal information application, PIA, and between two PIAs, using application to application (A2A) communication. A formal query rewriting algorithm is presented for the single PIA case.

In the future, we will continue the study and implementation of our framework. It is clear that a lot of the success of PIM systems lies on the successful automation of the different mechanisms that are needed. In particular, we will look further into the automation of the conceptualization of full-text files and that of matching resources to ontological concepts. Also, we will elaborate on the idea of 3D navigation both by studying a model for temporal navigation and by carrying out user studies. The study of A2A communication, including data exchange, collaboration, and query processing will also be continued. While RDQL queries are expressive, they may not be suitable for most users. We are therefore exploring visual queries that can express a class of RDQL queries "appropriate" for the semantic desktop.

References

1. M. Berland and E. Charniak. Finding Parts in Very Large Corpora. In *ACL*, 1999.

2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
3. P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.
4. V. Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, 1945.
5. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based Query Containment. In *PODS*, pages 56–67, 2003.
6. J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, 1987.
7. I. F. Cruz, H. Xiao, and F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. In *AP2PC*, July 2004.
8. S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. C. A. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.
9. X. Dong and A. Y. Halevy. A Platform for Personal Information Management and Integration. In *CIDR*, pages 119–130, 2005.
10. P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton. Extending Document Management Systems with User-specific Active Properties. *ACM Transaction of Information System*, 18(2):140–170, 2000.
11. S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. In *SIGIR*, pages 72–79, 2003.
12. E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, 1996.
13. J. Gemmell, G. Bell, R. Lueder, S. M. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex Vision. In *ACM Multimedia*, pages 235–238, 2002.
14. A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB J.*, 10(4):270–294, 2001.
15. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *WWW*, pages 556–567, 2003.
16. HP Labs. RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm, 2005.
17. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: the State of the Art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
18. M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, Madison, Wisconsin, June 2002. ACM.
19. D. Maier and L. M. L. Delcambre. Superimposed Information for the Internet. In *WebDB*, pages 1–9, 1999.
20. I. Mani. Recent Developments in Text Summarization. In *CIKM*, pages 529–531, 2001.
21. T. D. Millstein, A. Y. Halevy, and M. Friedman. Query Containment for Data Integration Systems. *Journal of Computer and System Sciences*, 66(1):20–39, 2003.
22. T. H. Nelson. Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-use. *ACM Computer Surveys*, 31(4es):33, 1999.
23. N. F. Noy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, 33(4):65–70, 2004.
24. D. Quan, D. Huynh, and D. R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *ISWC*, pages 738–753, 2003.
25. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
26. L. Sauermann. The Gnowsis Semantic Desktop for Information Integration. In *The 3rd Conference on Professional Knowledge Management*, pages 39–42, 2005.