A Layered Framework Supporting Personal Information Integration and Application Design for the Semantic Desktop *

Isabel F. Cruz Huiyong Xiao

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

Abstract. With the development of inexpensive storage devices, space usage is no longer a bottleneck for computer users. However, the increasingly large amount of personal information poses a critical problem to those users: traditional file organization in hierarchical directories may not be suited to the effective management of personal information because it ignores the semantic associations therein and bears no connection with the applications that users will run. To address such limitations, we present our vision of a *semantic desktop*, which relies on the use of ontologies to annotate and organize data and on the concept of *personal information application (PIA)*, which is associated with a user's task. The *PIA designer* is the tool that is provided for building a variety of PIAs consisting of views (e.g., text, list, table, graph), which are spatially arranged and display interrelated fragments of the overall personal information. The semantic organization of the data follows a layered architecture that models separately the personal information, the domain data, and the application data. The network of concepts that ensues from extensive annotation and explicit associations lends itself well to rich browsing capabilities and to the formulation of expressive database-like queries. These queries are also the basis for the interaction among views of the PIAs in the same desktop or in networked desktops. In the latter case, the concept of desktop service provides for a semantic platform for the integration of information across different desktops and the web. In this paper, we present in detail the semantic organization of the information, the overall system architecture and implementation aspects, queries and their processing, PIAs and the PIA designer, including usability studies on the designer, and the concepts of semantic navigation in a desktop and of interoperation in a network of desktops.

* This work was partially supported by NSF Awards ITR IIS-0326284 and IIS-0513553. A preliminary version of this paper was presented at the First International Workshop on the Semantic Desktop–Next Generation Personal Information Management and Collaboration Infrastructure, Galway, Ireland, November 2005 (in association with the International Semantic Web Conference): "A Multi-Ontology Approach for Personal Information Management," by Huiyong Xiao and Isabel F. Cruz).

1 Introduction

More than six decades after Vannevar Bush put forward Memex, his vision of a *personal information management* (*PIM*) system where semantic associations are emphasized [9], most computer users are still constrained by a rigid hierarchical file organization. To make matters worse, there is no alternate way, for example using different views, to access the personal information. In this paper, we describe a system that places a strong emphasis on data modeling to establish semantic associations and on interactive ways to view and manipulate the information and those associations. The supported data model enables expressive queries, which constitute the basis for the interactivity of the views in a local or networked environment.

Our work fits within the research on the *semantic desktop* (e.g., [54]), which leverages the potential of the semantic web [5] to address the challenges of complex personal information (PI) spaces. A *PI space* is composed of a wide variety of data, which are syntactically and semantically heterogeneous. The following simple scenario illustrates the heterogeneity of PI spaces and some of the challenges that make their management challenging.

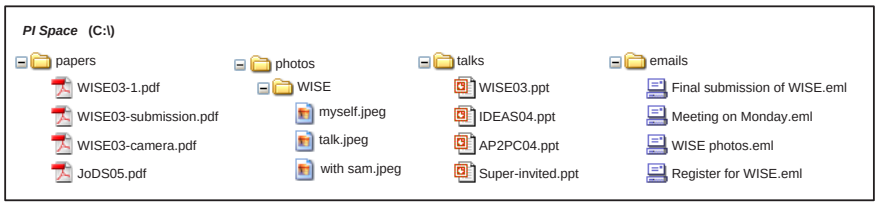

Fig. 1. An example of files in a desktop.

Example 1. Figure 1 presents a fragment of the personal information space that consists of four directories: papers, which contains a variety of papers, photos\WISE, which contains pictures taken at the WISE '03 conference, talks, which contains the files of four talks, and emails, which contains 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 may make it difficult to discover such relationships. Keyword-based search techniques, e.g., offered by the Google Desktop Search,[1] can retrieve all files that contain the word "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. Furthermore, the lack of semantic annotation will make it difficult to distinguish among files that are associated with the WISE

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

(Web Information Systems Engineering) conference and the WISE (Women in Science and Engineering) initiative—a case of semantic heterogeneity.

In the semantic desktop approach, as in the semantic web, a major role is played by ontologies that give meaning to data. Given the data associated with an application, also called *domain*, a *domain ontology* contains a conceptual description of that data, consisting of the *concepts* used, their *attributes*, and the *relationships* between those concepts. For example, the *Resource Description Framework*, *RDF*,[2] is based on *resources*, which correspond to concepts and on *properties* that correspond to attributes or to relationships between concepts. We further use the vocabulary language for RDF, the *RDF Schema (RDFS)*.[3] By using RDFS we can assert, for example, that a resource is a subclass of another resource or the range of a property. The semantic desktop vision consists of assuming that "all digital information items stored on a PC can be seen as web resources to which the RDF model can be applied." [54]. This vision also includes the concept of peer-to-peer (P2P) communication between applications in distributed desktops or within the same desktop. The advantage of this form of communication is that it is completely decentralized with all the applications being able to communicate with one another using the same protocol independently of where they reside.

Example 2. When modeling the Bibliography domain, an ontology will have resource Publication and its subclasses (also resources), which include Journal and Conference Proceedings. Publication has properties such as Title, Written by, and Year. The range of Written by is the resource Author, which in turn has several properties, including Name, whose range is the resource Literal.

In our paper, the term *semantic* denotes the meaning that is given to data by an ontology that annotates that data. For example, the meaning of the two WISE terms of Example 1 could be given by concepts in two different ontologies respectively describing conferences in Computer Science and women initiatives, or it could be given by different concepts in an ontology describing different organizations in Engineering. The idea of annotation to provide context is similar to the concept of *superimposed information*: "data 'placed over' existing information sources to help organize, access, connect and reuse information elements over those sources." [43]

We list next the objectives of our semantic desktop system and give a rationale for them.

Semantic data organization supporting a wide variety of associations, and a uniform representation. This organization provides several advantages. First, the annotations and associations provide context to the information, thus making data more easily understandable. Second, the superimposed information also allows for a finer and more expressive manipulation (e.g., querying) of the data. Third, an explicit formal semantics can facilitate reasoning on the data and

[2] http://www.w3.org/RDF/
[3] http://www.w3.org/TR/rdf-schema

deriving new knowledge. Finally, the uniform representation can support the integration of heterogeneous data.

Diversity of data manipulation methods encompassing the integration, exchange, browsing/navigation, and query processing of the stored personal information. The semantic data organization provides for a complex network of interrelated data on which expressive query languages can operate to retrieve the information that is related, to a task, project, or application. Likewise, browsing/navigation will be performed in such a network, instead of on hierarchies of folders. Interoperation among different information spaces in the same desktop or across different desktops and the web can be undertaken using, for example, the peer-to-peer (P2P) paradigm to enable communities of users to share information [23].

Multiple interactive visualizations for helping the user to understand data and their interconnections. Instead of providing separate data views as most traditional applications do, a semantic desktop should support data visualization from different perspectives, to offer a comprehensive view. Examples of visualizations of interest include association-centric visualizations, which are populated by queries [52], time-centric visualizations [31, 32], and views containing hierarchical lists of tasks provided as overlays over files, email, and web information [34, 35]. Furthermore, we want to provide a query-based mechanism that allows for two or more visualizations to interact, building on the semantic data organization and on the expressive (and formal) querying capabilities.

Our contributions in this paper are as follows:

1. We present a **layered framework** for the semantic desktop, in which multiple ontologies play a variety of roles. Specifically, the *resource layer* stores all the desktop resources (using URIs), metadata, 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 used to construct different ontologies for different user's applications. This layered architecture enables: i) a semantically-rich environment for personal information management; ii) a flexible and reusable system, by decoupling the domain and application ontologies, so that new domain ontologies that are becoming available in increasing numbers can be used to assemble new application ontologies. This layered architecture has advantages over the use of a single ontology that describes all the information in a desktop because such an ontology would likely: (1) not cover all the concepts of interest to all users; and (2) would not take advantage of the development of new domain ontologies.
2. We implement the concept of **superimposed information** for semantic organization, focusing on resource-file and resource-resource associations. We provide users with the capability to browse the network of annotated and interrelated information in the PI space in three different ways: within a layer, called *horizontal* navigation, across layers, called *vertical navigation*, and following their time stamps, called *temporal navigation*. This idea general-

izes approaches in other personal information and semantic desktop systems including MyLifeBits [32] and Placeless Documents [28].

3. We present the concept of **personal information application (PIA)**. Each PIA is an application that uses the personal information in the desktop for a specific purpose, such as for bibliography management, paper composition, or trip planning. For each of those purposes, there can be several applications. For example, for bibliography management there can be an application that manages and displays the references for a paper and another one that manages the user's Bibtex files. The PIAs can be stand-alone, with their own application ontology, user interface, and workflows. They can also communicate with one another 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.
4. We design the **architecture** of a system that supports our layered framework and the concept of superimposed information. There are three main large components in our architecture comprising: (i) the *data and metadata repositories*; (ii) the *semantic desktop server* consisting of metadata modules that extract and enhance the metadata of files (including file wrappers, an annotator, a classifier, and an indexer), design and matching ontology modules, and query processing modules; (iii) the *user interaction* modules for the design and browsing of the PIAs, respectively the *PIA designer* and the *PIA browser*, and for exploring resources (the *resource browser*).
5. We developed, implemented and user tested the **PIA designer** for end users to create PIAs. The PIA designer uses the MVC (Model-View-Controller) methodology [40] in assembling views and in defining their interaction within a PIA. We show how PIAs can use **desktop services** and how such services can facilitate data interoperation and integration across semantic desktops. We also developed several other components of our system.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we describe our layered framework and its main components. The semantic organization of the information (including the concepts of annotation, association, and representation) is discussed in Section 4. We present the system architecture and its main components and concepts, which include the concept of Personal Information Application (PIA) in Section 5. Query processing is discussed both in a single PIA and across PIAs in Section 6. Section 7 presents in detail the PIA development environment, which uses the MVC paradigm. Usability tests were performed in the PIA designer and their results are reported in Section 8. Semantic navigation is discussed in Section 9. In Section 10, we discuss both desktop interoperation as provided by desktop services and desktop service execution. In Section 11 we describe briefly the implementation of other components of the system. Finally, in Section 12 we summarize the paper and discuss future research directions.

2 Related Work

In response to the limitations currently imposed by desktops in what refers to PIM, an important body of research has emerged in recent years. This research addresses a wide variety of issues in areas ranging from data modeling, querying, presentation, and human-computer interaction to collaboration and information sharing, including privacy, security, and trust considerations [36]. The state-of-the-art of semantic desktop research has been described by Sauermann [55], while related research issues have been recently compiled [25, 26].

The Gnowsis semantic desktop project [54, 56] aims to develop a semantic desktop environment that supports P2P data management based on *desktop services*. As 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 the organization and manipulation of data, the latter being the focus of both SEMEX and of OntoPIM [14]. SEMEX is a personal data integration framework that uses a fine-grained annotation based on schemas, similar to our ontology-based framework [27]. However, a single domain model is provided as the unified interface for all data access, while we provide a layered framework using multiple ontologies to organize personal information. Our framework supports: (1) the use of a growing number of available ontologies that describe a wide variety of domains (e.g., email messages, bibliographic entries, or pictures) as standardized by others; and (2) the decoupling of the domain and application ontologies.

MyLifeBits [32], Haystack [52], and Placeless Documents [28] 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). 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. Placeless Documents introduces active properties, where documents can have associated applications that provide document-based services.

Stuff I've Seen (SIS) is based on a simple but powerful idea—people usually want to find information that they have already seen; therefore, it provides an indexed space that includes not only the information on a person's laptop, but also on visited web pages [29]. Lifestreams is based on a single metaphor for the storage, manipulation, and visualization of documents—a time-ordered stream of documents [31]. Project Xanadu has existed since 1960 and has its sources in hypertext [47]. Chandler[4] concentrates on information and communication tasks, such as composing and reading email, managing an appointment calendar

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

and keeping a contact list. Other interfaces for personal data management are based on Wikis and include SemperWiki [49] and WikSAR [1]. However, they resemble a hypertext composer (or content manager) providing the user with a means to put pieces of information together as a Wiki page.

Existing interfaces provide a workspace for the end user to develop applications. Such applications have their own data model, data presentation, and control logic. Of such interfaces, Haystack's end user interface [2] is the closest to the PIA designer that we introduce in this paper. Both interfaces support parameterizable channels that are collections of items retrieved by executing the channels (essentially queries). Their channel parametrization is oriented to individual channels, while ours can take the input from other channels so that two channels (with their associated views) can interact. Another difference is that in our framework, the channels and views are bound to *desktop services* defined in terms of PIAs, so that reusability is naturally implemented by desktop service composition.

The *Universal Labeler (UL)* presented by Jones *et al.* is a unified scheme for labeling all kinds of personal information (e.g., electronic documents, paper documents, email messages or web references) [34, 35]. It follows the principle that "Good information management follows from good project management." A project is a hierarchy of subprojects and tasks displayed in a window that not only manages the UL but also gives direct access to other applications (e.g., email or calendar). This hierarchy is created by a "drag-and-link" operation, in such a way that users drag only a fragment of a document to that hierarchy and the remaining information is hyperlinked to that fragment. Each node can own planning-oriented properties or behaviors, such as "remind me by" or due dates, that will be displayed, for example, in the user's calendar. A hierarchy provides therefore a view over the user's personal information. The UL approach does not attempt, however, to manage the storage of information items [35] like we do. In the UL approach, the work flow is driven by properties (e.g., time) or the order of the tasks, whereas we allow user-defined *if-then* rules that act on the views to control the work flow.

3 Framework

Our framework follows the principle of superimposed information, that is, of data or metadata "placed over" existing information sources [43]. This concept seems particularly useful for the organization, access, interconnection, and reuse of the information elements. We present a layered ontology-based framework, as shown in Figure 2, with the following 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 (for example, Bibtex entries). In contrast, complex-content files have

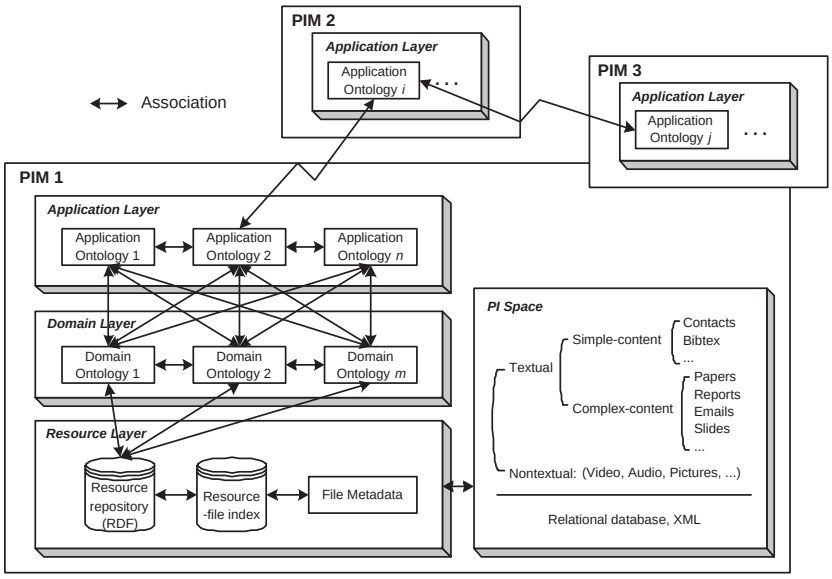

Fig. 2. An ontology-based framework for the semantic desktop. Three networked semantic desktops are represented, one of which is represented in detail.

a flexible scheme of presentation and may contain references to other files for example by means of citations or hypertext links [15], as is the case with papers that cite another paper (inside our outside the same desktop), which, in turn, could cite other papers.

File description. We annotate each file using metadata consisting of a file description containing 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 the DAML Ontology Library,[5] the Semantic Web Ontologies,[6] and the Protégé OWL ontologies.[7] The ontologies

[5] http://www.daml.org/ontologies/
[6] http://www.schemaweb.info
[7] http://protege.stanford.edu/plugins/owl/owl-library/

in these libraries are 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 files and 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*.

Application ontology. The *application layer*, which contains the ontologies for different applications, is superimposed on the *domain layer*. The domain ontologies of the domain layer enhance the reusability and flexibility of the framework, while the application ontologies are defined as views of the domain ontologies. In this way, the domain ontologies 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.

Besides the data components described above, a desktop 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), *file wrappers* (for identifying and extracting resources from files), and an *ontology designer* (for importing and editing an ontology).

4 Semantic Data Organization

In this section, we discuss in detail the mechanisms that our framework uses to support the semantic organization of the personal information space, including those for semantic annotation, association, and representation.

4.1 Annotation

Because the personal information space contains the base information, all the other data components in our framework are superimposed information over this space. The most fundamental function of the superimposed information is to provide semantic annotations of the base information. We discuss the following two aspects:

File description. Both textual and non-textual files will contain a description of their contents to enable keyword searches. For example, in non-textual files, the submitted keywords (e.g., "Canon") or key-value pairs (e.g., "Maker:Canon")

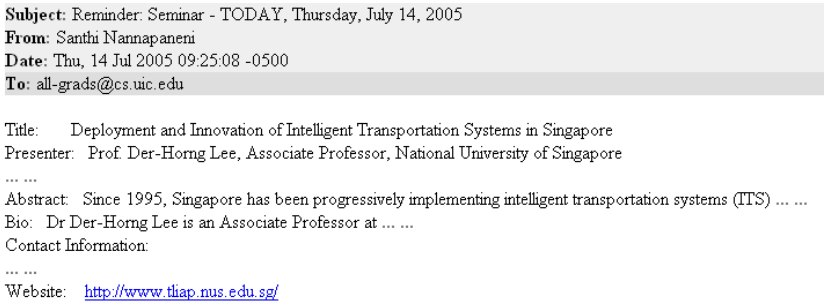

Fig. 3. Example of an email message.

will be matched to the property-value pairs of the file description, to find the right files requested by the user.

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 precise data retrieval. We note that a file can be an instance of more than one concept, according to different classification criteria.

4.2 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*, which are the resource-file indexes and the *resource-resource associations*, which 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 3. 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 properties a new instance (i.e., resource), which is associated with the saved email message 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.[8]
- **Hyperlink analysis**. For the textual files that include hyperlinks to classified resources (e.g., a citation of a paper or a link to a web page), 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., [4]) to parse each sentence of a text or its summary obtained by means of text summarization [44]. 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.

Resource-resource associations. We borrow from object-oriented design 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 personal information space, and classify them into the domain ontologies, thus populating the ontologies. In the example of Figure 3, it is possible to extract a pattern $\langle$Singapore, implements, ITS$\rangle$, which can then be classified as an instance of an ontological pattern such as

[8] 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		√		√	√

⟨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 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 [37].

4.3 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). For the schema of these data (i.e., the application and domain ontologies), we use the vocabulary language for RDF, RDF Schema (RDFS). 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) of the form $\langle subject, predicate, object \rangle$. RDFS is used to define the vocabulary (in terms of classes and properties) of 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 personal information is represented as a set of interrelated resources. In contrast, XML is not chosen because it cannot represent semantic associations [24]. 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 described next 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 [20]. For the representation of the *instantiation* and *generalization* relationships, we use rdf:type and rdfs:subClassOf, respectively.

Table 2. RDF properties for the associations.

Relationship	RDF property	Semantic description
aggregation	rdfx:contains	<#a, rdfx:contains, #b> means that a contains b.
property	User-defined properties	<#wise03talk, presentedBy, #xiao> means that wise03talk is connected to xiao by the association presentedBy.
instantiation	rdf:type	<#xiao, rdf:type, #Person> means that the resource xiao is an instance of the concept Person.
generalization	rdfs:subClassOf for classes and rdfs:subPropertyOf for property	<#Book, rdfs:subClassOf, #Publication> means that the concept Book is a subclass of the concept Publication.

The *property* relationship is represented naturally by an RDF property defined in the user-defined namespace.

Figure 4 gives a concrete example of the RDF representation of the application, domain, and resource layers. Two application ontologies for PIAs (picture management and publication management) are constructed from four domain ontologies (Email, Talk, Publication, and Photo). In the resource layer, the resource-file and resource-resource associations are represented as triples or in a graph.

5 System Architecture

Figure 5 presents the architecture of a system that implements our framework. We describe the primary components of the architecture, including the components related to semantic data organization (on the server side) and those related to user interaction (on the client side).

5.1 Semantic Data Organization

Our framework goes beyond a traditional hierarchical file organization by means of two types of ontologies: domain ontologies and application ontologies. The former represent the conceptualization of different domains, thus providing a foundation for personal data classification. The latter are designed to serve as the data model underlying personal information applications (PIAs), which are developed by end users. More details of how these ontologies cooperate to enable a semantically powerful data manipulation in the semantic desktop are given in Section 7.

File wrappers. The semantic organization is based on the analysis and processing of text documents in the personal information space. That is, we do not consider the non-textual features of a file, although such features may facilitate data annotation [8]. A *file wrapper* is used to retrieve text from various types of files, such as PDF, PPT, and DOC. The other functionality of file wrappers

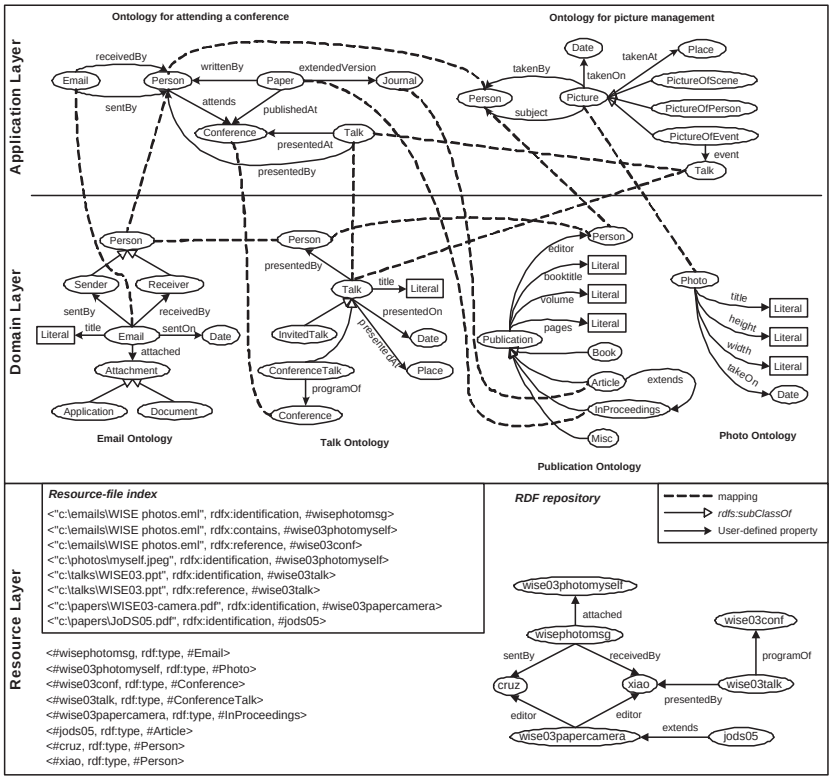

Fig. 4. Representation of the application, domain, and resource layers. All ontologies are represented in RDFS. Two application ontologies for PIAs, i.e., picture management and publication management, are constructed. Below them are four ontologies for the domains of Email, Talk, Publication, and Photo,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

is to obtain from the file system the system-defined properties of a file, e.g., its MIME type, size, and date.

Annotator. The annotator is responsible for creating and enhancing the annotation (or metadata) of a file. It is fed with the results of file wrappers, including the retrieved text and its standard properties, based on which it associates the file with property-value pairs. Most of current data annotators need input from users, although sometimes part of the annotations can be obtained from the file content. In practice, a semi-automatic annotator is often provided, such as the "easy" annotation mechanism of MyLifeBits [32]. The annotations are stored in a database, called *file description*.

Classifier. The classifier is one of the most important components for the semantic organization in the framework. Given a file and its file description, the

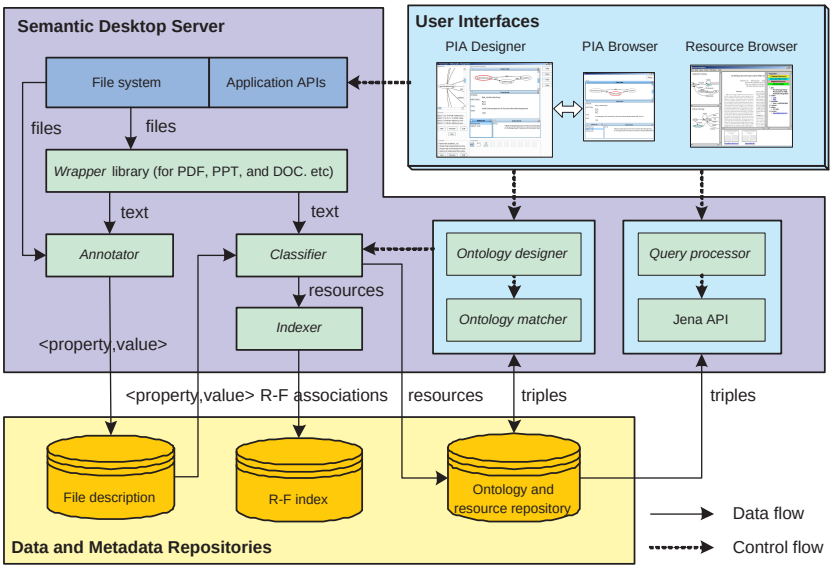

Fig. 5. Semantic desktop architecture.

classifier provides the following operations: (1) Identification of the file as a resource with a unique URI (Universal Resource Identifier); (2) Examination of the file content to explore the resources that are *contained* or *referred to* by the file; (3) Population of domain ontologies with all discovered resources; (4) Determination of the associations between resources, called *resource-resource (R-R) associations*. These resources and their associations are maintained in a *resource repository*.

Indexer. After being classified, a file is indexed in terms of the resources discovered in itself (e.g., the names of the authors in a publication). Such resource-file indices are stored in a repository, called *R-F index*, to be used in query answering. There are three types of R-F indices (also called R-F associations): *identification*, *containment*, and *reference*, which are obtained by the first and second operations of the classifier. Given a query of keywords posed by the user, the query processor can first locate the corresponding resources and then find the files that are identified as, containing, or referring to such resources, by means of the R-F index.

Ontology designer and matcher. At the center of the architecture are the multiple application and domain ontologies stored in the *ontology repository*. We provide an *ontology designer* for the management of concepts and roles of individual ontologies, and an *ontology matcher* for the maintenance of inter-ontology relationships (i.e., ontology mappings). Considering that most semantic desktop end users may lack the knowledge of particular ontology languages (e.g., RDFS or OWL), the ontology designer should hide the details of such languages but

enable users to work with the conceptualization of their domains of interest. In addition, to improve the precision of an automatic ontology mapping process, the ontology matcher may be able to combine different ontology matching strategies [18, 37]. Using recent advances in ontology matching, high precision and recall can be obtained even for large ontologies (a few thousand concepts) [30, 58]

5.2 Semantic Data Manipulation

To retrieve relevant data from the PI space, the user's request may be posed as a sequence of keywords or as a query language expression.

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 *ranked* results [53]. The results of a search are usually evaluated using 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 the exact match between the query and the data, so that recall is influenced because some relevant, yet unmatched data, is not retrieved. The two approaches complement each other, therefore it is desirable to provide both of them. In Section 6 we focus on query processing.

5.3 User Interaction

Based on the semantic data organization using multiple ontologies, the architecture comprises the following user interfaces to manipulate personal information: a desktop-wide browser called *resource browser*, and two other user interfaces, the *PIA designer* and *PIA browser*, for PIA development and execution, respectively. The functionalities of the PIA designer and browser are described in Section 7. The resource browser is described in Section 9.

6 Semantic Query Processing

In this section we discuss query processing in two cases: within a PIA and across different PIAs.

6.1 Query Processing in a PIA

We express our queries in RDQL (RDF Data Query Language) [33]. RDQL has an SQL-like syntax. The most important component of an RDQL query is the *Triple Pattern Clause* that matches the triples in the RDF database. Because of the implementation, we committed early on to RDQL, but a similar framework can be used for query languages such as RQL [38] or SPARQL.[9]

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

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 to play the same role as the global ontology (since the user query is posed on it), the whole system can be seen as a *Global as View (GaV)* data integration system [41]. In a GaV based integration system, query processing is performed using a "unfolding" strategy [41]. In particular, when the user poses a query over the application ontology, it gets rewritten into a new RDQL query in terms of the domain ontologies, using the mappings between the global ontology and the domain ontologies. These mappings are expressed as RDF class or property correspondences. The rewritten query is executed on the corresponding domain ontologies and resources that match the query are returned as answers.

We show in Figure 6 the query rewriting algorithm on a single PIA, which we call ADRewriting (for Application to Domain ontology Rewriting). For simplicity, we assume that the user queries are formulated in a subset of RDQL that we call *conjunctive RDQL (c-RDQL)*, which can be expressed as a conjunctive formula: $ans(\boldsymbol{X}) \text{ :- } p_1(\boldsymbol{X_1}), ..., p_n(\boldsymbol{X_n})$, where $\boldsymbol{X_i} = (x_i, x_i')$ and p_i is an RDF property of x_i with value x_i'. To rewrite 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 We assume that there are no integrity constraints over the application ontologies and do not show namespaces for simplicity.

Example 4. Suppose that the user wants to list all conference papers with their respective authors and journal version, using the query

$q_1 : ans(x, y, z) \text{ :- } writtenBy(x, y), extendedVersion(x, z).$

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: $ans(x, y, z) \text{ :- } editor(x, y), extends(z, x).$

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

6.2 A2A Query Processing

Application to application (A2A) query processing occurs when an application needs to retrieve relevant data from another semantically related application in order to answer a query. If the PIAs are seen a connected peers (i.e., service providers for a particular data access), A2A query processing is similar to that that occurs in peer-to-peer (P2P) systems [10, 61, 62]. The PIAs may exist in a single desktop or may be physically distributed across different desktops.

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. $q_2 = head_{q_2}$:- $body_{q_2}$;

Fig. 6. The ADREWRITING algorithm for application to domain ontology rewriting.

The A2A query processing consists of two steps. 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 $\mathcal{G}_2$, using 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 latter step is described by the algorithm of Figure 6, whereas the former step differs slightly from that algorithm in that instead of a 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* [6, 48] and *reasoning on queries* [11, 45]) have been extensively studied and are beyond the scope of this paper.

7 Personal Information Applications

In this section we describe in detail the *PIA designer*, a visual environment that supports the development of a PIA.

7.1 MVC-based PIA Development

The resource explorer allows for the "global" exploration of the resources and ontologies in a desktop. However, views need to be tailorable to the users' diverse tasks.

Each PIA can work in a stand-alone mode, with its own application ontology, user interface, and workflows, aiming at a specific task (e.g., bibliography management, paper composing, or trip planning). Meanwhile, different PIAs can communicate with each other as in a P2P network, by means of the connections (mappings) established between their application ontologies. A PIA can present two modes: *development mode* and *execution mode*. The interfaces corresponding to these two modes are respectively the *PIA designer* (for the development mode) and the *PIA browser* (for the execution mode), which can be switched from one to the other anytime.

The development of a PIA uses the MVC (Model-View-Controller) methodology. In particular, in the development of a PIA, the "Model" can be an application ontology that has been composed as a view over domain ontologies; the "View" consists of one or more components that present data in different forms such as graph, text, and list; the "Controller", which is the business logic of the PIA, is a set of "if-then" rules, which enable the interaction and synchronization between different data components. The data associated with components to be displayed are retrieved from the repositories of ontologies and instances by queries named *parameterized channels*.

The specifications of a PIA, as defined by the user by means of the PIA designer, including the model, view, and business logic, can be serialized in XML. It is called the PIA definition. A user can run a PIA in the PIA browser, which interprets and executes the PIA in either an "online" mode (by directly switching from the designer to the browser) or an offline mode (by loading from the PIA's permanent serialization). The separation of the declarative specifications from the interpretative execution greatly benefits the communication between semantic desktops in terms of PIA interoperation, as we will see in the following sections.

7.2 Implementation

We have implemented a prototype of the PIA designer, whose user interface is shown in Figure 7. We describe next the three stages needed for the desing of a PIA.

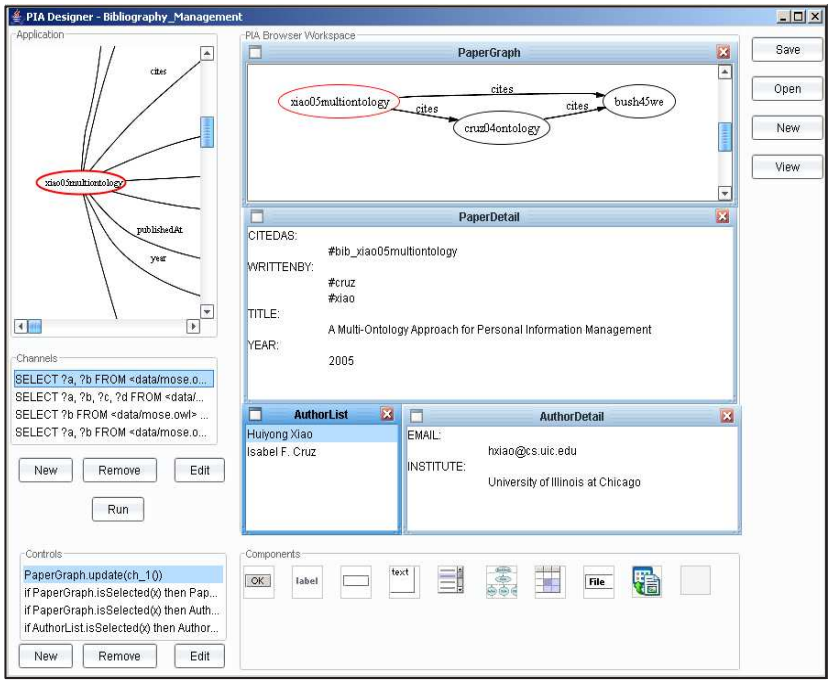

Fig. 7. The PIA designer. The PIA browser being designed contains the PaperGraph, PaperDetail, AuthorList and AuthorDetail panes

.

Modeling. In the first stage, the user loads the application ontology from the ontology repository, which represents the model underlying the PIA to be designed; it will be graphically shown in the *Data Model* pane. The application ontology is mapped to the domain ontologies, under which the resources representing personal information are classified. Actually, the application ontology is constructed as a view over the domain ontologies in a GaV approach. This mapping process should not require the users' programming expertise, but only their awareness of the task and their knowledge of the domain.

Visualization. The second stage involves the design of the layout of the PIA, with one or more visual components, each of which can be associated with a stream of data for its presentation. The user drags the desired visual components from the *Visual Component* pane to the *PIA Browser Workspace* pane. Examples of such components include TextPane, List, Table, Graph, and File. The associated data can be resources, strings, files, and whatever as instances of the ontologies; they are retrieved by queries, called *channels* (introduced in [52]), on the application ontology. Some components, such as Button, Label, TextInput, and MessageBox, are used to facilitate the interaction between the user and the

PIA browser. A special component called Services is used for desktop service composition, as discussed in Section 10.

Controller and parameterized channels. In the final stage, the controller (or business logic) of a PIA is specified so as to realize rich interactions between the data and their views, and to synchronize several visualizations. These controllers manage all possible updates of the model and handle the events from the user interface, using "if-then" rules (more sophisticated controls will be considered in future work) of the following form:

if $\texttt{Component}_1.\texttt{event}_1(x_1)$ **and** ... **and** $\texttt{Component}_n.\texttt{event}_n(x_n)$
then $\texttt{Component}_1.\texttt{action}_1(y_1)$; ...; $\texttt{Component}_m.\texttt{action}_m(y_m)$;
endif

where x_i, $i \in [1..n]$, are parameters passed from the events, and y_i, $i \in [1..m]$, are the *channels* that result in the actions. It often happens that the response of a component to some event needs to take x_i as a parameter to execute y_i, especially when updating the data that is sensitive to x_i in a visual component. For this purpose, we introduce the concept of *parameterized channel*, which are channels that have their contents determined by the parameters at runtime. Channels are queries over ontologies and the parameter of a channel can be bound to a variable or a constant in the query. By means of parameterized channels, an event started from a component can pass any values to another component, thus enabling interactions between different components.

Example 5. As shown in Figure 7, at the top left corner, the user loads the application ontology (for publications), to develop a PIA for bibliography management. The application's user interface uses a Graph for displaying the citation network of papers, a TextPane for the paper's details, a List for the paper's authors, and a TextPane for the author's details.

To associate data with their proper visualization, the user defines the following channels using the syntax of RDQL. Each channel is in the form of string, which can then be fed into an RDQL interpreter (e.g., provided by the Jena API) for execution.

1. **ch_1()**: *"SELECT ?a, ?b WHERE (?a, cites, ?b)"*
2. **ch_2(**x**)**: *"SELECT ?a, ?b, ?c, ?d WHERE (" + x + ", title, ?a), (" + x + ", writtenBy, ?b), (" + x + ", year, ?c), (" + x + ", citedAs, ?d)"*
3. **ch_3(**x**)**: *"SELECT ?a WHERE (" + x + ", writtenBy, ?b), (?b, name, ?a)"*
4. **ch_4(**x**)**: *"SELECT ?a, ?b WHERE (" + x + ", institute, ?a), (" + x + ", email, ?b)"*

As an example of parameterized channel, the second query, $ch\_2(x)$, returns the title, author, year, and citation entry of a publication, which is bound to parameter x.

The data computed by executing a channel will present different forms depending on what visual component is used to visualize this data. For example,

a Graph shows the data represented as a graph, where nodes are resources and edges are their associations. To construct such a graph, the nodes representing the same resource will be merged into a single one.

The following rules specify the controller. The first rule has no preconditions, thus being triggered at the very beginning of the PIA's run.

1. *PaperGraph.update(ch_1())*
2. ***if*** *PaperGraph.isSelected(x)* ***then*** *PaperDetail.update (ch_2(x))*
3. ***if*** *PaperGraph.isSelected(x)* ***then*** *AuthorList.update (ch_3(x))*
4. ***if*** *AuthorList.isSelected(x)* ***then*** *AuthorDetail.update (ch_4(x))*

8 Evaluation of the PIA Designer

We undertook usability studies on our prototype of the PIA designer. Given the current implementation of the prototype where users must compose queries and controls, the kind of knowledge needed corresponds to that of CS majors. Therefore, all the 33 participants were Computer Science Majors at UIC, including 19 undergraduate and 14 graduate students.

The evaluations took place in two desktop computers in our research lab. Prior to executing their tasks, all participants were given a lecture (one hour and 15 minutes) covering the following topics: conceptual models, including the RDF model, and its query language, RDQL. Of this lecture, a slide described the organization of personal information in current desktops (Figure 1) and the fact that the Semantic Desktop is "a conceptual approach to represent, manipulate, and access Personal Information in a desktop". Another slide showed the main screen of the PIA designer. The syntax of the controls was presented in a single slide. Finally, a two minute demo was given of the PIA designer prototype. Although this was not intentional, the demo was given by a graduate student who is not part of the prototype implementation team. Likewise, the lab evaluations were administered and then compiled and summarized by three students (two graduates and one undergraduate) who were also not part of the PIA implementation team. This fact was mentioned to the participants who knew from the beginning that their evaluations and answers would not be judged by students directly connected with this particular project. Except for two participants in the study, none of the participants of the study had prior knowledge of conceptual models or of query languages (such as SQL).

The participants in our study were given a list of tasks to execute. To verify that all the tasks were executed, questions were asked in which the participants had to write the outcome of each task. As a consequence, there was not a fixed time for their participation but most of the participants finished the tasks in less than 25 minutes. There were a couple of outliers: a participant who finished the tasks in 10 minutes and another one that took almost 35 minutes.

The participants had three tasks to perform. In the first one, they browsed through the domain ontology and identified concepts in the graph. In the second task, they had to create two views and their associated queries and controls. They were not asked to create the queries from scratch because it was felt that

testing the users on that particular subtask would address the usability of RDQL itself not that of the prototype. Also, the prototype does not currently support debugging of the RDQL syntax. Therefore, their understanding of syntax of the query language was tested in a separate written question. Finally, the participants were presented with a complex screen with 4 views and had to execute a sequence of tasks to test their understanding of the interactions among views.

After completing their tasks, the participants filled out a questionnaire containing 41 questions divided into 6 parts: *I. Past experience* (with desktop search and browsing, email management, and their predictions on manual vs. automatic annotation of documents), *II. Overall reactions to the prototype*, *III. Screen organization and manipulation*, *IV. Learning*, *V. Queries and controls*, and *VI. Final comments*. Of the 41 questions, 7 questions asked for textual answers in free format and the remaining questions were multiple choice, with the choices ranging from 1 (low) to 5 (high). Part VI contained a single question to be answered textually. It was meant to allow for overall comments that the participants might not have expressed before. The participants completed the questionnaire in 15-20 minutes. Again, an outlier took just over 5 minutes to complete the questionnaire, whereas some of the participants took as much as 25 to 30 minutes to complete the questionnaire.

In their past experience (Part I of the questionnaire), most participants were Windows users. As for ways to find information in their desktops, the participants prefer browsing the contents of their hard drive (3.8/5) rather than conducting a search (2.6/5). Their comments regarding search as provided by windows state "too slow, too many results," "output not organized." They were also generally happy with searching for email messages and their attachments using their email program (3.5/5). This answer is highly correlated with how often they keep their messages and attachments in their mailbox instead of saving them (3.5/5). When asked if they would annotate each of their documents manually if annotating one document would take about 30 seconds they seemed in general to be willing to do so (3.2/5). They were, however, more enthusiastic about letting the computer classify their information automatically (4.1/5) even if they would not trust that classification completely (2.8/5). Their trust increased substantially if they were allowed to see the classification schema used by the computer (3.9/5). A table that shows the averages and standard deviation for the answers to the questions for Part I is shown in Figure 8.

We subdivided the overall reactions to the prototype (Part II of the questionnaire) in five categories: *overall experience*, *overall concept*, *ease of use*, *power*, and *implementation effort needed for broader deployment*. Ease of use was given a positive assessment (3.5/5), while the overall experience was rated slightly lower (3.2/5). The score given to the amount of implementation still needed was 3.5/5. We found that the overall experience is correlated with the score attributed to the graph layout and its manipulation in Part III of the questionnaire (respectively 3.3/5 and 3.4/5). Incidentally, the task associated with the graph layout was also the one where the participants spent most time. The graph layout ca-

Question	Measure	Category	Average	Std Dev
Q1.1	frequency	desktop search	2.8	1.3
Q1.2	satisfaction	desktop search	2.7	1.3
Q1.5	frequency	desktop search if satysfying	3.6	1.3
Q1.7	frequency	email with info to search later	3.8	1.1
Q1.8	satisfaction	email search	3.5	1.2
Q1.9	frequency	keep messages in inbox	3.5	1.5
Q1.10	satisfaction	folder organization	3.6	1.2
Q1.11	ease	desktop browsing	3.8	1.2
Q1.12	frequency	manual classification if available	3.2	1.1
Q1.13	frequency	automatic classification if available	4.1	0.9
Q1.14	trust	automatic classification if available	2.8	0.9
Q1.15	trust	browsable automatic classification if available	3.9	1.0

Fig. 8. Summary of the questions and answers (average and standard deviation) of Part I.

pability is provided by a third party.[10] For large graphs, we do not anticipate a simple solution to the difficulties encountered by the study participants except that we believe that with some training users would be able to better read the graph.

The highest scores went to overall concept and to power (both with 3.8/5). Free form comments from the participants point mainly to the difficulty of the tasks associated with the graph layout and its manipulation: "I didn't figure out which text label went with certain edges. Zooming in didn't help." "Highlighting the nodes and edges was difficult."

Free form comments from the participants regarding the overall interface were in general very encouraging, especially considering that the participants received no training on the prototype:

"It was simple to understand and very intuitive." "I could trial and error it with a little bit of effort." "Basically it's easy to use and also it's such a powerful application to show the concept." "Good to use but not sure about overall capability. Since it is a prototype, hard to learn or use. Immediate updates on multiple views are very good." "New prototype; never used before. Took some getting used to."

A table that shows the averages and standard deviation for the answers to the questions for Part II is shown in Figure 9.

The participants' feedback on the screen layout and interaction (Part III of the questionnaire) was requested on the overall screen layout (3.5/5), button placement (3.5/5), switching between screens and going to the next screen (3.8/5), and progression of work related tasks (3.8/5). Feedback on the graph layout was also collected, which covered two aspects: readability of the graph (3.3/5) and interaction with the display (3.4/5). Higher satisfaction was reported by the participants in what concerns building the view layout (4/5) and entering/editing the queries and controls (4.2/5).

[10] www.graphviz.org

Question	Measure	Category	Average	Std Dev
Q2.1	satisfaction	overall experience	3.2	0.8
Q2.2	satisfaction	overall concept	3.8	0.9
Q2.3	ease	use	3.5	0.9
Q2.4	adequacy	power	3.8	0.8
Q2.5	amount	development work left	3.5	1.0

Fig. 9. Summary of the questions and answers (average and standard deviation) of Part II.

In their free form comments, participants mentioned some problems with the screen layout: "Components are not aligned well together. Tool buttons on bottom seem not usual (better on top). Enter/edit window is too small to see the whole text." "Everything seemed to flow well, except I had to stop and think when adding components as they only had images" "The screen was pretty easy to navigate. Would like the option to click on buttons, instead of drag and drop." "Drag and drop is always good. The channels and controls text field should be bigger with a better explanation of what to enter." "Window size of the PIA browser workspace could be bigger or more optimal as side menus could be made smaller." "Could use some colour, otherwise very good." "To the point and I like that you don't have to use multiple screens for one thing."

A table that shows the averages and standard deviation for the answers to the questions for Part III is shown in Figure 10.

Question	Measure	Category	Average	Std Dev
Q3.1	helpfulness	screen layout	3.5	0.8
Q3.2	helpfulness	button placement	3.5	1.1
Q3.3	clarity	switching screens	3.8	0.9
Q3.4	clarity	task progression	3.5	0.8
Q3.5	clarity	graph layout	3.3	1.1
Q3.6	clarity	interaction with graph	3.4	1.0
Q3.7	ease	build view layout	4.0	0.8
Q3.8	ease	edit channels and controls	4.2	1.0

Fig. 10. Summary of the questions and answers (average and standard deviation) of Part III.

The answers of the participants on their experience with learning the prototype (Part IV of the questionnaire), was subdivided into 4 questions: getting started (3.6/5), learning advanced features (3.1/5), exploration of features (3.5/5), and their appreciation on whether the steps to achieve a task follow a logical sequence (3.5/5). The comments were positive: "Was simple to understand and very intuitive." "It was easy to learn the application and exciting as well." "It was easy once you got the hang of it. The window should provide some definition of the terms." "Most things fairly easy to navigate, self-explanatory."

"It was easy enough but I could see someone who isn't computer savvy get confused." "Learning the system was a good experience."

A table that shows the averages and standard deviation for the answers to the questions for Part IV is shown in Figure 11.

Question	Measure	Category	Average	Std Dev
Q4.1	ease	getting started	3.6	0.9
Q4.2	ease	learning advanced features	3.1	0.7
Q4.3	ease	exploration of features	3.5	0.9
Q4.4	clarity	sequence of task steps	3.5	0.8

Fig. 11. Summary of the questions and answers (average and standard deviation) of Part IV.

Finally, the participants were asked about the functionality of the prototype in what concerns queries and controls (Part V of the questionnaire). The participants were asked to rate the ease of understanding the interaction between queries and controls (3.7/5) and of expressing RDQL queries (3.6/5). The participants were asked about their appreciation of the necessary skills to express RDQL queries and controls. In a range from 1 to 5, where 1 refers to *casual user* and 5 to expert user, the scores leaned to the latter (respectively 3.6/5 and 3.4/5). The appreciation of the statement "With some training, a computer science major would define new RDQL queries or new controls" with 1 corresponding to *with difficulty* and 5 to *easily* leaned toward the latter (4.3/5).

The following answers are illustrative of the comments obtained: "It was nice to see how a query would bring up a graph and use the controls to define how content appears." "Seems difficult if you do not have complete understanding of RDF. Also the query making should be automated because of syntax issues." "'The queries were a little guess work but easy, the control implementation was slightly confusing." "It was easy at first and then harder." "It was practical but could be more user friendly." "For a CS graduate it will be easy to use within a few minutes to the full extent of the application. For a novice, queries and understanding controls is difficult." "Still I need to understand how queries interact with controls." "The queries are a little confusing, but not terribly hard."

The two participants with SQL experience, wrote: "Much easier than the SQL control features used at work. Query statements just need to get used to." "Some specifics to RDQL need to be re-learned when coming from a history of using MySQL queries–yet they are similar enough that the learning curve is not particularly steep."

A table that shows the averages and standard deviation for the answers to the questions for Part V is shown in Figure 12.

Most participants offered additional comments in Section IV:

"I don't see it being widely used for home users. Company users would love it. Lots of time people work together but don't classify data in the same way, even though they are talking about the same thing."

Question	Measure	Category	Average	Std Dev
Q5.1	expert level	writing RDQL queries	3.6	1.0
Q5.2	expert level	control writing	3.4	0.8
Q5.3	ease	writing queries and controls for CS majors	4.3	0.7
Q5.4	clarity	interaction between queries and controls	3.7	0.8

Fig. 12. Summary of the questions and answers (average and standard deviation) of Part V.

This insightful observation came from an undergraduate. We note that in introducing the project, the topic of semantic heterogeneity was not mentioned.

Other participants summarized their overall experience or emphasized their previous suggestions for improvement:

"I like how I can view all information about documents by a series of clicks. I like the layout as well. I would love to see a demo of how the software can be used to organize all the documents on a PC." "Very cool prototype. I like it." "As mentioned before, the idea is amazing. Make it user friendly with menus, query making (get info from user and generate a statement)." "Requires some learning time." "Overall the PIA designer has a good concept that needs a few minor touches to make before becoming a product for the casual user." "Just like common software applications, good to have menu bar on top and tool set right below. In graph view, in addition to scroll bar, it would be very nice to have a panning/zooming function function with mouse move." "There should be a default view with the controls that would help any novice users. A tool tip mouse over would help give the user feedback on what the buttons did." "Great concept, add highlighting of the arrow labels on the ontology view."

9 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.

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 the data. Compared to the traditional navigation approach that is based on hierarchical directories, this semantic navigation is based on semantic associations, similarly to those that humans establish between concepts. More specifically, by taking into account the layered architecture, the semantic navigation in our framework can be performed in three directions: (1) Using *vertical navigation*, the user follows a path across layers. Two cases are possible for this kind of navigation: top-down from

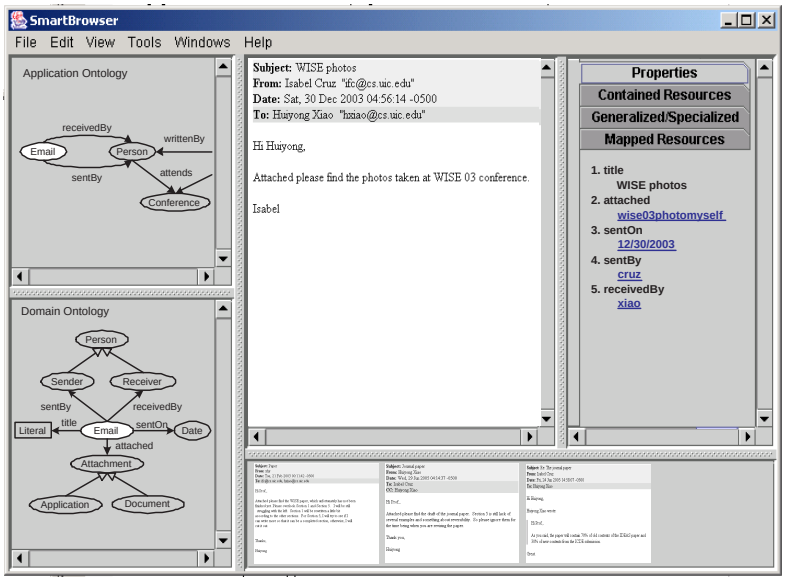

Fig. 13. The resource browser.

the application ontologies to the stored files and bottom-up from the stored files to the application ontologies. (2) Using *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) Using *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.

Example 6. Consider the scenario shown in Figure 4 for an example of semantic navigation. Suppose the user selects the Email node in the application ontology for conference attendance. Following the vertical associations (i.e., mappings) from the application ontology and the Email domain ontology, we can show to the user the context of the Email concept, including its title, sender and receiver, and date sent. Further in this direction, all Email instances (e.g., wisephotomsg representing an email message with some photos taken at the WISE '03 conference attached), as well as the actual email messages represented by these instances, can be located and shown to the user.

If the users want to know more about the sender A of the email message, such as the talks given by A they can switch the context from the Email domain ontology to the Talk domain ontology by horizontally following the mapping between the Person concepts in both ontologies). Further navigation from the Talk ontology to the Publication ontology would enable users to browse the publications authored by A.

Given that the sentOn property of the Email concept is of type Date, the email resources, as instances of the Email concept, can then be sorted by the date sent, so that the user can go over the email messages in chronological order.

The resource browser that is shown in Figure 13 implements the semantic navigation described above. The domain ontology shows the context of whatever concept selected in the application ontology. The bottom-right pane shows a timeline of different resources belonging to the current concept. Those resources can be sorted chronologically based on the time stamps of the objects that those resources represent (e.g., the email files). The central pane shows the object represented by the current resource, to the right of which there is a tabbed pane containing the resources associated with the current resource. These associated resources are categorized to guide the user in the semantic navigation process. In particular, the Properties tab pane shows the values of all the properties present in the context (domain ontology) of the current resource. The other three tab panes show different types of ontology mappings connected to the current resource: Contained Resources that aggregates the resources under the current resource; Generalized/Specialized that contains those superconcept/subconcept resources of the current resource; Mapped Resources contains the resources that are mapped to the current one.

10 Interoperation and P2P architecture

A networked semantic desktop framework such as the one in Figure 2 would allow for the interoperation among collaborating semantic desktops. For example, in a book writing project, authors would be able to share their different bibliographies even if they are in a variety of formats. As mentioned before, two PIAs can communicate in a P2P fashion based on the application ontology mappings established between them. There have been plenty of previous work on central and P2P ontology and data integration [21, 22, 60, 63]. However, it is required in this case that the two PIAs are designed for a similar task, for which they have their application ontologies overlapping partially or fully. This case was covered in detail in Section 6.

In this section, we discuss another type of PIA interoperation, which is realized by means of desktop services instead of by the engineering of ontology mapping, thus called *service-oriented interoperation*. The notion of *desktop service* was first introduced by the Gnowsis system [54]. In our case, we want to integrate the notion of desktop service with that of building visualizations in a PIA and consequently with channels.

In general, a service must have its interface (i.e., input and output) defined, while keeping the implementation of its operation hidden from the service consumer. A PIA consists of a set of visual components bound to parameterized channels. We can, therefore, view a channel as the minimal unit of service taking parameters as input and having data (e.g., sets of tuples) as output. We support both a flexible composition of desktop services and a flexible construction of visualizations that uses a controller consisting of *if-then rules* to specify

the composition (control and data flows) among channels and desktop services in a PIA. Types of service composition include sequential and parallel flows [50].

For example, consider the PIA of Example 5 (see also Figure 7) as a desktop service provider. The service architect can specify each of its four channels as a service, for example, ch_1() returns all pairs of publication citations, ch_2() outputs the details of a publication given as input, ch_3() outputs the authors of a publication supplied as input, and finally ch_4() outputs the details of a given author. Besides, the scheduler can choose to parallelize ch_2() and ch_3() (since they accept the same kind of input) to provide a single service. Or, it can serialize ch_2() and ch_4() as a service outputing detailed information about the authors of a publication, given that ch_4() can take as input one of ch_3()'s output (i.e., the author's resource id).

A practical issue relates to the execution of desktop services for which we identify two cases. The first case is about the *remote execution* of desktop services. In the example, there are four services (PIA-1 to PIA-4), with their respective application ontologies (AO-1 to AO-4). Suppose that PIA-4 is the starting point of the service execution, where the user interacts with the PIA browser. All requests for the data and for the execution of other services (defined and implemented in other desktops, but composed by the current service) are driven by events from such interactions. Whenever a nested remote service (e.g., PIA-2 or PIA-3) is triggered by the current service, a request for execution will be sent to the respective remote desktop (e.g., SemDesk 2 or SemDesk 3), where the remote service will be executed. As a response to the request, the remote service returns its execution results to the current service.

While the first case is similar to what happens with web services, the second case of desktop service execution, which we call *local execution*, is quite different. In particular, whenever a service nested in the current service is activated, it will be locally interpreted and executed by the PIA browser in the current desktop. However, the local execution of a remote service (e.g., PIA-2) needs permission to access relevant data (e.g., described by AO-2) from a remote desktop. If so, the data is then duplicated in the local desktop via a secure data transfer.

The essential difference between these two cases illustrates the tradeoff between control permission and data access. This flexibility is important in a semantic desktop setting. Depending on their available resources, some desktops may be reluctant to take a heavy workload while some others may be concerned with the privacy of their data. Therefore, a desktop (when acting as a server) can choose whether to contribute its computing power or share its data.

11 Implementation

In addition to the PIA designer that was described in Section 7, we have implemented several of the components of our semantic desktop, which we describe next.

The semantic annotator implements the file wrapper and semantic annotation described in Section 5. Annotation is a semi-automatic process, which combines

a template-based metadata extractor with the user's manual input. Currently, the system has included several commonly used templates, which are designed based on domain ontologies, such as email, photo, movie, paper, and talk. Using this annotator over a corpus of 150 files showed that as many as 70% of the fields in the various templates are automatically extracted by the annotator.

The current version of the data classifier is directed to text classification. Considering only hierarchical ontologies under which files will be categorized, our classifier applies a hierarchical Bayesian classification method based on the work of Koller and Sahami [39], which is shown to be better than flat classification, in terms of both precision and recall.

The visual query generator provides a visual interface for the user to view RDF ontologies as graphs and to easily construct (RDQL) queries, also visualized as graphs, over the ontologies. The interface uses a drag-and-drop feature for forming conjunctive ontology queries, simply by dragging graph objects (nodes or edges) from the ontology panel to the query panel. Zooming is also provided in case the user wants to observe the graph details. The interface treats queries as first-class objects amenable to different kinds of manipulations, including renaming, deletion and reuse.

Figure 14 shows the user interface of the visual query generator. A list of ontologies is displayed on the top left-hand side of the user interface. The RDF graph representing the selected ontology is displayed on the top right-hand side. On the bottom half of the user interface, from left to right, the following displays can be seen: a list of saved queries, the visual and textual representation of the query being edited, and a pane showing the results of the current query displayed as a graph. The user can examine this graph as a way to determine the correctness of the query. This interface, which is currently stand-alone, will be integrated with the PIA designer to facilitate the formulation of the RDQL queries.

The semantic data visualizer uses a layered approach that adds presentation specifications to desktop data using two concepts called Semantic Lens and Presentation Lens by extending our previous work and the work by others [3, 17, 51]. The visualizer can filter and aggregate data stored in RDF format. The presentation specifications provide different users with the ability to visualize data using different paradigms (e.g., timelines, bar charts, graphs) according to their own preferences and the characteristics of the data. In particular, there is a layer that extracts essential metadata to produce an appropriate display. For example, an ordered domain can be represented with certain visual attributes (e.g., length) whereas a non-ordered domain can be represented with other visual attributes (e.g, color) [7]. Instances that have a property with values in the temporal domain can be displayed using a time line as shown in Figure 15. This visualizer will be integrated with both the Resource Explorer (see Section 9) and with the PIA designer to allow for views that aggregate information.

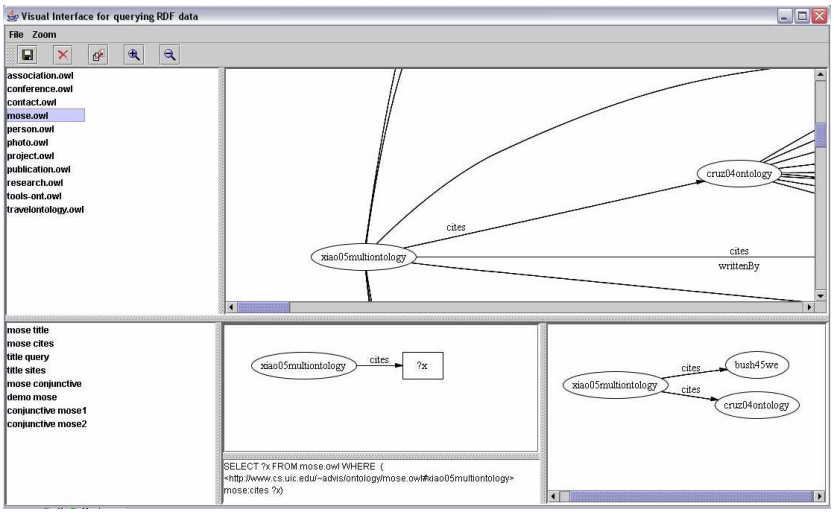

Fig. 14. Visual query interface to express RDQL queries.

12 Conclusions and Future Work

As new paradigms for data organization and manipulation become available, such as those provided by the semantic web, it is natural to explore ways in which those paradigms can be used to address shortcomings in the management of highly-heterogeneous personal information. In this paper, we have considered the use of ontologies and associated query languages to manage personal information in a desktop.

In particular, we have devised an ontology-based framework consisting of three loosely-coupled layers: the application layer, the domain layer, and the resource layer. The first layer supports directly the specific user's activities, such as travel planning to attend a conference. The ontology that models the application provides a view over the ontologies at the domain level. Those ontologies are not specialized to meet the requirements of a particular user, but are standardized and engineered to meet wide use requirements in applications such as air travel, hotel booking, or event registration. At the resource layer, we have addressed the organization of data including the use of file descriptions and domain ontologies as annotations and the extensive use of data associations.

The extensive annotation and the associations among data create a network of data and metadata that: (1) can be traversed using the concept of semantic navigation; and (2) lends itself well to the formulation of expressive database-like queries. To replace hierarchical browsing and keyword querying in today's desktop, we have proposed respectively the concepts of semantic navigation, which enables users to follow the associations among data at each layer and across layers and database-like querying. Database-like querying is also closely associated with the concept of personal information application, PIA, which

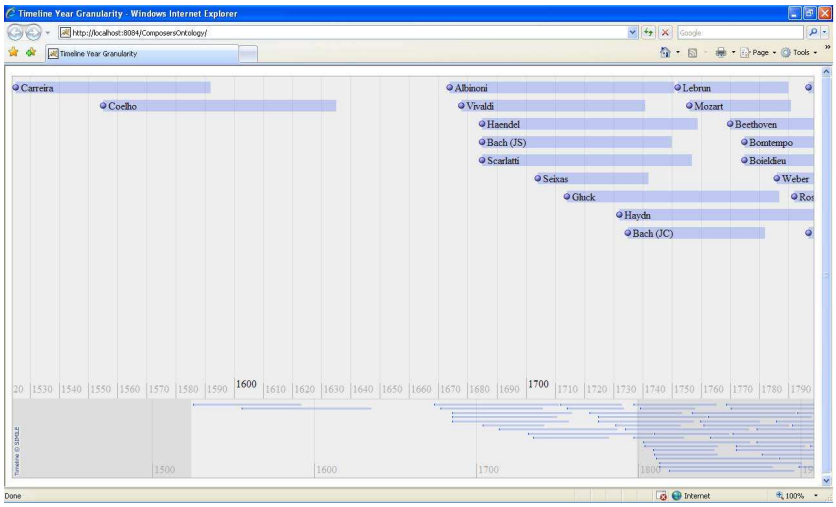

Fig. 15. Visual display of an RDF file using a timeline lens.

is used to provide users with interactive views of the information of interest and to emphasize the associations therein. Queries are used to select both the information to display and as channels to express the interaction among views.

We have devised a query rewriting algorithm for conjunctive queries in a single PIA. This technique can be extended to application-to-application (A2A) query processing in the same desktop or across different desktops, using mappings between the corresponding ontologies. Interoperation across applications can also be achieved by using the concept of desktop services that extends the notion of channels to bridge across different PIAs.

The specification of a PIA is performed using a tool called the PIA designer, with which the users define the displays, the queries, and the channels. We have conducted user studies to evaluate our prototype of the PIA designer by computer science majors (albeit with limited knowledge of data modeling or query languages) and received positive feedback. It was, however, recognized by the participants that queries and channels are more suitable for expert users than for casual users. We have also undertaken the implementation of several of the architectural components of our system.

In the future, we will pay special attention to the usability of the PIA designer and of the other components of our system so as to enable casual users to access more of its functionality. An area to address is that of query languages [12]. In particular, visual query [13, 16] and keyword-based languages will be investigated as well as context-aware information search [59], which could significantly help users in formulating simpler, yet precise queries. Likewise, we will be considering natural language specifications to automatically formulate channels; in this context, previous work on the conversion of natural language questions to

formal queries is of interest [42]. Another issue relates to the personalization of the domain of interest using the user's profile as in OntoPIM [14].

We will also continue our work and the work of others [54, 56] on mechanisms for defining, publishing, discovering, and composing desktop services so as to extend their current capabilities. We will also address research issues related to the minimization of the manual intervention that is required to build the semantically-rich network of information, including the automation of the classification, annotation, and discovery of data associations.

Finally, we will investigate security and privacy issues that arise when sharing data across multiple semantic desktops, including, for example, the issue of privacy-preserving ontology matching [19, 46, 57].

Acknowledgements

We would like to thank Ryan Aviles for his help with the implementation of the PIA designer. We are thankful to Wenyuan Fei, Rigel Gjomemo, and Brian Rhoades for their help with the design and administration of the usability studies. We also owe thanks to Abhishek Gurunathan, Seshank Kalvala, and Fang Alice Wang for the design and implementation of the visualizer, to Akshay Nadkarni for his work on text classification, to Ramprasad Chandrasekaran for the implementation of the visual query interface, and to Swati Tata and Srikant Vemuri for their work on the annotator.

References

1. D. Aumueller and S. Auer. Towards a Semantic Wiki Experience – Desktop Integration and Interactivity in WikSAR. In *ISWC Workshop on the Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure*, 2005.
2. K. Bakshi and D. R. Karger. End-User Application Development for the Semantic Web. In *ISWC Workshop on the Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure*, pages 123–137, 2005.
3. B. B. Bederson and J. Meyer. Implementing a Zooming User Interface: Experience Building Pad++. *Software: Practice and Experience*, 28(10):1101–1135, 1998.
4. M. Berland and E. Charniak. Finding Parts in Very Large Corpora. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 1999.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, pages 29–37, May 2001.
6. P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.
7. J. Bertin. *Semiology of Graphics*. The University of Wisconsin Press, Madison, Wisconsin, 1983.
8. S. Bloehdorn, K. Petridis, C. Saathoff, N. Simou, V. Tzouvaras, Y. S. Avrithis, S. Handschuh, I. Kompatsiaris, S. Staab, and M. G. Strintzis. Semantic Annotation of Images and Videos for Multimedia Analysis. In *European Semantic Web Conference (ESWC)*, pages 592–607, 2005.
9. V. Bush. As We May Think. *The Atlantic Monthly*, 176(1):101–108, 1945.

10. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. What to Ask to a Peer: Ontology-based Query Reformulation. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 469–478, 2004.
11. D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. View-based Query Containment. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 56–67, 2003.
12. T. Catarci. What Happened When Database Researchers Met Usability. *Information Systems*, 25(3):177–212, 2000.
13. T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini. Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing*, 8:215–260, 1997.
14. T. Catarci, L. Dong, A. Halevy, and A. Poggi. Structure Everything. In W. Jones and J. Teevan, editors, *Personal Information Management*. University of Washington Press, 2007.
15. J. Conklin. Hypertext: An Introduction and Survey. *IEEE Computer*, 20(9):17–41, 1987.
16. I. F. Cruz. DOODLE: A Visual Language for Object-Oriented Databases. In *ACM SIGMOD International Conference on Management of Data*, pages 71–80, 1992.
17. I. F. Cruz and Y. F. Huang. A Layered Architecture for the Exploration of Heterogeneous Information Using Coordinated Views. In *IEEE Symp. on Visual Languages and Human-Centric Computing*, pages 11–18, 2004.
18. I. F. Cruz, W. Sunna, and A. Chaudhry. Semi-Automatic Ontology Alignment for Geospatial Data Integration. In *International Conference on Geographic Information Science (GIScience)*, volume 3234 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2004.
19. I. F. Cruz, R. Tamassia, and D. Yao. Privacy-Preserving Schema Matching Using Mutual Information. In *Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, volume 4602 of *Lecture Notes in Computer Science*, pages 93–94. Springer, 2007.
20. I. F. Cruz, H. Xiao, and F. Hsu. An Ontology-based Framework for Semantic Interoperability between XML Sources. In *International Database Applications and Engineering Symposium (IDEAS)*, pages 217–226, July 2004.
21. I. F. Cruz, H. Xiao, and F. Hsu. An Ontology-based Framework for Semantic Interoperability between XML Sources. In *8th Int. Database Engineering and Applications Symposium (IDEAS)*, pages 217–226, 2004.
22. I. F. Cruz, H. Xiao, and F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. In *Third Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*, volume 3601 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2005.
23. S. Decker and M. R. Frank. The Networked Semantic Desktop. In C. Bussler, S. Decker, D. Schwabe, and O. Pastor, editors, *WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web*, volume 105 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
24. 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.
25. S. Decker, J. Park, D. Quan, and L. Sauermann, editors. *ISWC Workshop on the Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure*, volume 175 of *CEUR Workshop Proceedings*. ISWC Workshop, CEUR-WS.org, November 2005.

26. S. Decker, J. Park, L. Sauermann, S. Auer, and S. Handschuch, editors. *Semantic Desktop and Social Semantic Collaboration Workshop (SemDesk) located at the International Semantic Web Conference (ISWC)*, volume 202 of *CEUR Workshop Proceedings*. ISWC Workshop, CEUR-WS.org, November 2005.
27. X. Dong and A. Y. Halevy. A Platform for Personal Information Management and Integration. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 119–130, 2005.
28. 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.
29. 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 *International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 72–79, 2003.
30. J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Šváb, V. Svátek, W. R. van Hage, and M. Yatskevich. First Results of the Ontology Evaluation Initiative 2007. In *Second ISWC International Workshop on Ontology Matching*. CEUR-WS, 2007.
31. E. Freeman and D. Gelernter. Lifestreams: A Storage Model for Personal Data. *SIGMOD Record*, 25(1):80–86, 1996.
32. J. Gemmell, G. Bell, R. Lueder, S. M. Drucker, and C. Wong. MyLifeBits: Fulfilling the Memex Vision. In *ACM International Conference on Multimedia*, pages 235–238, 2002.
33. HP Labs. RDQL - RDF Data Query Language. `http://www.hpl.hp.com/semweb/rdql.htm`, 2005.
34. W. Jones, H. Bruce, A. Foxley, and C. F. Munat. Planning Personal Projects and Organizing Personal Information. In *Annual Meeting of the American Association for Information Science and Technology (ASIS&T)*, 2006.
35. W. Jones, C. F. Munat, H. Bruce, and A. Foxley. The Universal Labeler: Plan the Project and Let Your Information Follow. In *Annual Meeting of the American Association for Information Science and Technology (ASIS&T)*, 2005.
36. W. Jones and J. Teevan, editors. *Personal Information Management*. University of Washington Press, 2007.
37. Y. Kalfoglou and M. Schorlemmer. Ontology Mapping: the State of the Art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.
38. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A declarative query language for RDF. In *International World Wide Web Conference (WWW)*, pages 592–603, 2002.
39. D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *International Conference on Machine Learning (ICML)*.
40. G. E. Kramer and S. T. Pope. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 1(3):26–49, August/September 1988.
41. M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.
42. Y. Li, H. Yang, and H. V. Jagadish. NaLIX: an Interactive Natural Language Interface for Querying XML. In *ACM SIGMOD International Conference on Management of Data*, pages 900–902, 2005.

43. D. Maier and L. M. L. Delcambre. Superimposed Information for the Internet. In *ACM SIGMOD Workshop on The Web and Databases (WebDB)*, pages 1–9, 1999.
44. I. Mani. Recent Developments in Text Summarization. In *International Conference on Information and Knowledge Management (CIKM)*, pages 529–531, 2001.
45. 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.
46. P. Mitra, C.-C. Pan, P. Liu, and V. Atluri. Privacy-preserving Semantic Interoperation and Access Control of Heterogeneous Databases. In *ACM Conference on Computer and Communications Security*, pages 66–77, 2006.
47. 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.
48. N. F. Noy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, 33(4):65–70, 2004.
49. E. Oren. SemperWiki: a Semantic Personal Wiki. In *ISWC Workshop on the Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure*, 2005.
50. C. Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, 2003.
51. E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel: A browser-independent presentation vocabulary for rdf. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2006.
52. D. Quan, D. Huynh, and D. R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. In *International Semantic Web Conference (ISWC)*, pages 738–753, 2003.
53. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
54. L. Sauermann. The Gnowsis Semantic Desktop for Information Integration. In *3rd Conference on Professional Knowledge Management*, pages 39–42, 2005.
55. L. Sauermann, A. Bernardi, and A. Dengel. Overview and Outlook on the Semantic Desktop. In *ISWC Workshop on the Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure*, 2005.
56. L. Sauermann, G. A. Grimnes, M. Kiesel, C. Fluit, H. Maus, D. Heim, D. Nadeem, B. Horak, and A. Dengel. Semantic Desktop 2.0: The Gnowsis Experience. In *International Semantic Web Conference (ISWC)*, volume Volume 4273, pages 887–900. Springer, 2006.
57. M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy Preserving Schema and Data Matching. In *ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2007.
58. W. Sunna and I. F. Cruz. Using the AgreementMaker to Align Ontologies for the OAEI Campaign 2007. In *Second ISWC International Workshop on Ontology Matching*. CEUR-WS, 2007.
59. J. Teevan, R. Capra, and M. Pèrez-Quiñones. How People Find Information. In W. Jones and J. Teevan, editors, *Personal Information Management*. University of Washington Press, 2007.
60. H. Xiao and I. F. Cruz. RDF-based Metadata Management in Peer-to-Peer Systems. In *The 2nd IST Workshop on Metadata Management in Grid and P2P System (MMGPS)*, 2004.

61. H. Xiao and I. F. Cruz. Integrating and Exchanging XML Data Using Ontologies. In *Journal on Data Semantics VI*, volume 4090 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2006.
62. H. Xiao and I. F. Cruz. Ontology-based Query Rewriting in Peer-to-Peer Networks. In *2nd International Conference on Knowledge Engineering and Decision Support (ICKEDS)*, pages 11–18, 2006.
63. H. Xiao, I. F. Cruz, and F. Hsu. Semantic Mappings for the Integration of XML and RDF Sources. In *VLDB Workshop on Information Integration on the Web (IIWeb)*, pages 40–45, 2004. `http://cips.eas.asu.edu/iiwebfinalproceedings/22.pdf`.