Prototipo per la rappresentazione di semantici web services e la loro discovery basata sui dati (Report 4.1.4)

Sommario: questo documento è stato realizzato nell'ambito dell'attività OR 4.1

Autore: M. Vincini, M. Orsini, S. Bergamaschi, L. Sgaravato, , M. Interlandi, F. Guerra

Stato Finale
File:
Data: 20/07/2011 16.15

Diffusione		
	X	solo tra partner
		tra partner e partecipanti focus group
		pubblica

Introduction

From a user perspective, data and services provide a complementary vision of an information source: data provide detailed information about specific needs, while services execute processes involving data and returning an informative result too. For this reason, users need to perform aggregated searches able to identify not only relevant data, but also services able to operate on them. At the current state of the art such aggregated search can be performed manually only by expert users, that first identify relevant data, and then identify existing relevant services.
This report describes a semantic approach to perform aggregated search of data and services. In particular, we developed a technique that, on the basis of an ontological representation of data and services related to a domain, supports the translation of a data query into a service discovery process.
In particular we designed and developed a system architecture able to create a semantic peer data ontology (SPDO) representing the common knowledge extracted from heterogeneous sources, and an information retrieval-based web service engine (XIRE) able to provide a list of ranked services according to a set of weighted keywords. Starting from an SQLlike query expressed in the terminology of the SPDO, and thanks to the extraction algorithm we propose, our systems is able to retrieve all data and services that are relevant to the query. The aggregated search is achieved by:

(i) building the SPDO,
(ii) building a Global Light Service Ontology (GLSO) consisting of the lightweight version of the ontologies used in the service semantic descriptions
(iii) defining a set of mappings between the SPDO and the GLSO
(iv) exploiting, at query time, term rewriting techniques based on these mappings to build a keyword-based query for service retrieval express in the GLSO terminology starting from a SQL query on the data sources.

Preliminary evaluations based on state-of-the-art benchmarks for semantic web services discovery show that our information retrieval-based approach provide promising results.

Motivating scenario and Running Example

Let us assume that a virtual touristic district is composed of a set of touristic companies (including travel agency, hotels, local public administrations, touristic promotion agencies) creating a semantic peer in which they want to share and expose an integrated view of touristic data and services.
The semantic peer wants to supply the tourist promoters and travelers with all the information about a location by means of only one tool managing both data and services provided by different web sources. Let us introduce as an example three information systems about Italian locations that may be integrated to create a larger information source available for touristic purposes:
• BookAtMe provides information about more than 30.000 hotels located in more than 8.000 destinations. For each hotel, information is provided about facilities, prices, policies, etc. Some services are also available for checking the availability of a room and booking it.
• Touring provides information about Italian hotels, restaurants and cities. By means of three different forms, it is possible to find the available hotels, restaurants (described with some specific features) and main monuments for each city.
• TicketOne provides information about Italian cultural events. For each event, a description including place, price and details is provided. The information system offers services to check the ticket availability of a particular event and, also, to buy tickets.
A global view of the data sources provided by each of these information systems is created by means of a data integration system and shown in Figure 1. To give an idea of the global view obtained, hotels and restaurants are located in a certain city. A restaurant can be either the restaurant of a certain hotel or not. Every hotel has certain facilities, and some kind of activities can be performed as the facilities of that hotel. Events take place in a city, and the hotels offer vacation packages for specific events. Information about customers of the hotels and about bookings made by customers for certain hotels are stored. Moreover, cars are available for rental in certain cities.
Now let us consider that a user wants to find the name of hotels available in Modena.

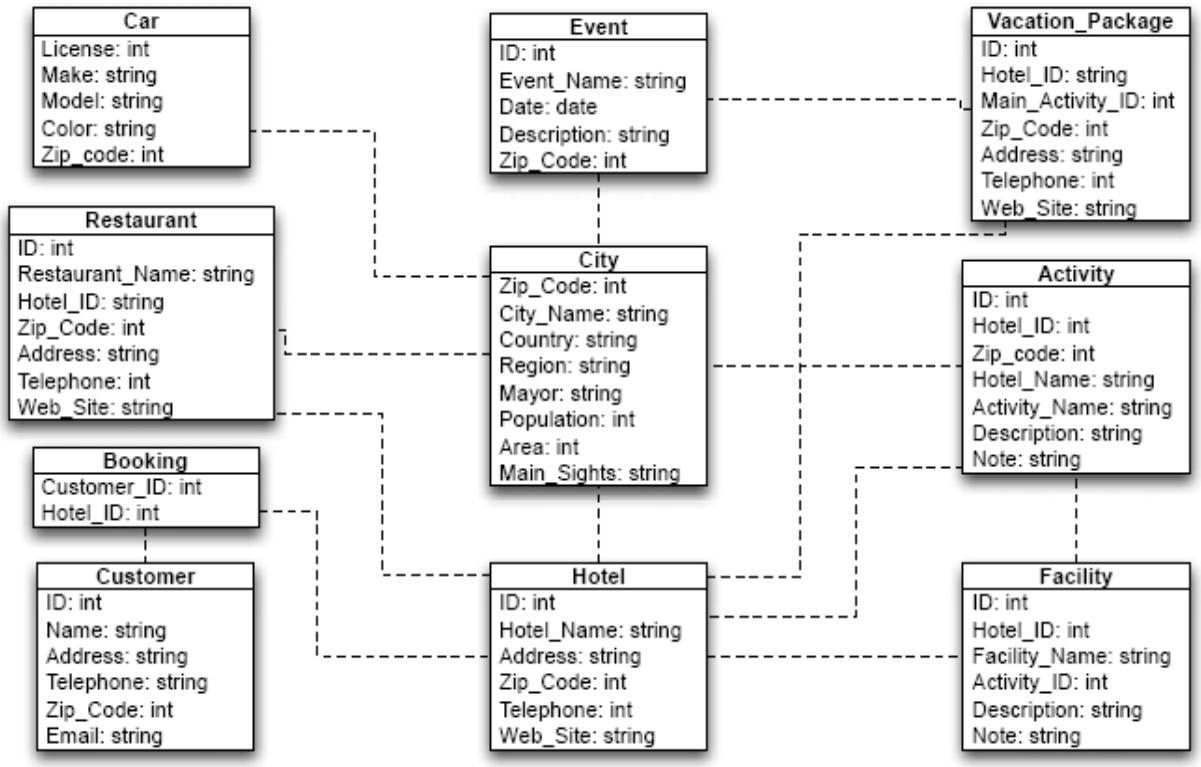

Figure 1 - The tourist integrated schema

select Hotel.*
from Hotel
where Hotel.City = 'Modena'

While the problem of finding relevant data for such query is well defined, an important problem is to retrieve, among the many services available and related to the several domains mentioned above, the ones that are possibly related to Q1, according to the semantics of the terms involved in the query.

Building the Global Data and Service View at Set-up Time

In our approach, data sources and services are grouped into semantic peers. Each semantic peer generates a Peer Virtual View (PVV), i.e. a unified representation of the data and the services held by the sources belonging to the peer. A PVV is made up of the following components, shown in Figure 2:

- a Semantic Peer Data Ontology (SPDO) of the data, i.e. a common representation of all the data sources belonging to the peer; the SPDO is built by means of MOMIS.
- a Global Light Service Ontology (GLSO) that provides, by means of a set of concepts and attributes, a global view of all the concepts and attributes used for the descriptions of the Web services available in the peer;
- a set of mappings which connect GLSO elements to SPDO elements.

Building an Integrated Representation of Data: the SPDO

The SPDO is a virtual integrated view of the underlying sources for which mapping rules and integrity constraints are specified to handle heterogeneity, created using the MOMIS system. The process is based on clustering techniques applied to a set of metadata collected in a Common Thesaurus (CT) in the form of relationships describing inter- and intra-schema knowledge about classes and attributes of the local data source schemas. These relationships are extracted from descriptions of local schemas, obtained from the relationships existing in the WordNet database between the meanings associated to the source elements by a semi-automatic process, and inferred by means of Description Logics techniques.
Several languages have been developed for representing data sources especially due to the growth of the sources available on the Internet that has required the definition of languages for sharing data. Our choice is to use and extend an object-oriented language as it offers some accurate capabilities for source descriptions. The SPDO is then represented by means of the ODLI3 language, which is an extension of the ODL language, an object oriented language developed by ODMG1. ODLI3 is transparently translated into a Description Logic. ODLI3 allows different kinds of data sources and the view resulting from the integration process to be represented in a common data model. In ODLI3 all the data sources are represented as a set of classes and attributes. Moreover, some constructors and rules are present in the language to handle the data heterogeneity:

Union constructor. The union constructor is introduced to express alternative data structures in the definition of an ODLI3 class, thus capturing requirements of semistructured data.

Optional constructor. The optional constructor is introduced for class attributesto specify that an attribute is optional for an instance (i.e., it could be not specified in the instance).

Integrity constraint rules. This kind of rule is introduced in ODLI3 in order to express, in a declarative way, if then integrity constraint rules at both intra- and inter-source level.

Intensional relationships. They are terminological relationships expressing intra- and inter-schema knowledge for the source schemas. Intensional relationships are defined between classes and attributes, and are specified by considering class/attribute names, called terms. The following relationships can be specified in ODLI3:

- syn (Synonym-of), defined between two terms ti and tj, with ti = tj , that are considered synonyms in every considered source (i.e., ti and tj can be indifferently used in every source to denote a certain concept).
- bt (Broader Terms), or hyperonymy, defined between two terms ti and tj such as ti has a broader, more general meaning than tj . bt relationship is not symmetric. The opposite of bt is nt (Narrower Terms), or hyponymy.
- rt (Related Terms), or positive association, defined between two terms ti and tj that are generally used together in the same context in the considered sources.

An intensional relationships is only a terminological relationship, with no implications on the extension or compatibility of the structure (domain) of the two involved classes (attributes).
Extensional relationships. Intensional relationships syn, bt and nt between two classes C1 and C2 may be "strengthened" by establishing that they are also extensional relationships.

Mapping Rules. This kind of rule is introduced in ODLI3 in order to express relationships holding between the integrated ODLI3 schema description of the information sources and the ODLI3 schema description of the original sources.

Using ODLI3 for representing sources and ontologies is not a limitation: the interoperability of the ODLI3 descriptions is guaranteed by a software module able to translate such descriptions into the languages for describing sources and ontologies on the web, i.e. OWL, RDF, XML (Schema).

The MOMIS integration process for building the SPDO has five phases:

1. Local source schemata extraction. Wrappers analyze sources in order to extract (or generate if the source is not structured) schemas. Such schemas are then translated into the common language ODLI3.

2. Local source annotation with WordNet. The system, exploiting different algorithms[5], automatically suggests a meaning for each element of a local source schema, according to the WordNet lexical ontology. The integration designer is supported by a GUI in reviewing and, if necessary, correcting the proposed annotations. MOMIS also allows the user to extend the WordNet ontology by adding new concepts and relating them to the native elements of WordNet.

3. Common thesaurus generation. Starting from the annotated local schema, MOMIS constructs a set of relationships describing integrated intra-schema knowledge about classes and attributes of the source schemata. The CT is incrementally built by starting from schema-derived relationships, i.e. automatic extraction of intra-schema relationships from each schema separately. Then, the relationships existing in the WordNet3 database between the annotated meanings are exploited by generating relationships between the respective elements that are called lexicon-derived relationships. The Integration Designer may add new relationships to capture specific domain knowledge, and finally, by means of a Description Logics reasoner, ODB-Tools, which performs equivalence and subsumption computation, the MOMIS system infers new relationships and computes the transitive closure of CT relationship.

4. SPDO generation. The MOMIS methodology, applied to the CT and the local schemata descriptions, generates an affinity matrix exploiting the similarity measure of the elements of the sources. A hierarchical clustering technique applied to this affinity matrix groups similar elements of different sources, exploited for generating a global schema and sets of mappings with local schemata. The Global Schema is made up of a set of global classes. Several global attributes belong to a global class.

5. SPDO annotation. By exploiting the annotated local schemata and the mappings between local and global schemas, the MOMIS system assigns semi-automatically a name and a meaning to each element of the global schema. The result is thus an ontology made of a set of Global Classes, each composed of Global Attributes, that is a representation of the knowledge provided by the data sources available in the peer.

To give a brief example, let us consider three local sources:

L1.CUSTOMER (ID, NAME, ADDRESS, TELEPHONE),
L2.CLIENT (ID, NAME, ADDRESS, ZIP CODE),
L3.USER (ID, USERNAME, EMAIL).
These three local classes are represented in ODLI3as:

Source L1
interface customer {
attribute id integer,
attribute name string,
attribute address string,
attribute telephone integer
}

Source L2
interface client {
attribute id integer,
attribute name string,
attribute address string,
attribute zip_code integer
}

Source L3
interface user {
attribute id integer,
attribute username string,
attribute email string,
}

In the annotation phase, the element L1.customer is annotated with the WordNet synset someone who pays for goods or services. The same annotation is suggested for the element L2.client. According to WordNet, these two terms are synonyms. Thus, the relationship existing in theWordNet database leads to the definition in the Common Thesaurus of the SYN relationship between the two elements customer and client. The element L3.user is then annotated with the synset a person who makes use of a thing; someone who uses or employs something. Since customer and client are hyponyms of user, in the CT two NT relationships are generated between customer and user, and between client and user. Being very similar, these three local classes are grouped together by the clustering algorithm, generating the global class Customer. This global class is composed of the union of the attributes coming from the three local sources it is mapped on. The global class is then represented in ODLI3as:

```
interface customer {
attribute id integer,
attribute name string,
attribute address string,
attribute zip_code integer,
attribute telephone integer
attribute email string,
}
```

Finally, the global class is automatically annotated starting from the annotations provided to the local classes that compose it and considering the mappings between the local and the global schemas.

GLSO Construction and Semantic Similarity Matrix

The global light service ontology is built by means of a process consisting of three main steps: (i) service indexing, (ii) Global Service Ontology (GSO) construction, (iii) Global Light Service Ontology (GLSO) construction and Semantic Similarity Matrix (SSM) definition. A sketch of the overall process is given in Figure 2.

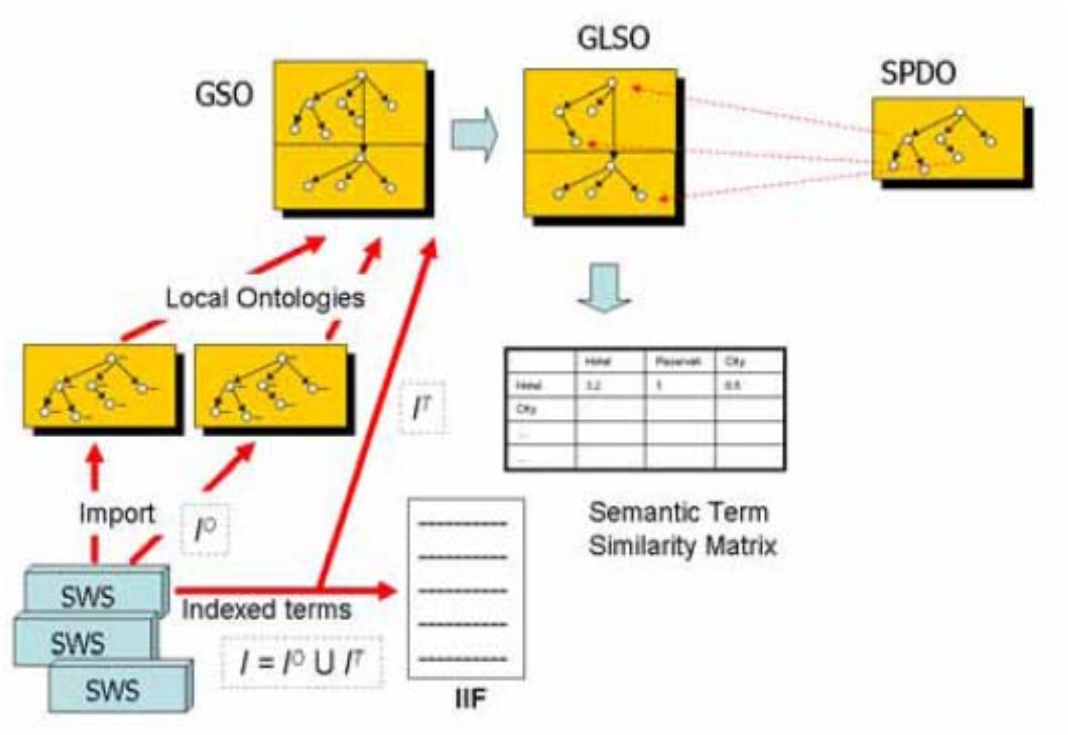

Figure 2: A sketch of the overall process for building the GLSO

Service Indexing

In order to inquiry services, an Information Retrieval approach is applied to the semantic descriptions of Web services. We consider OWL-S as the semantic Web Service description language where OWL-S semantic descriptions are OWL ontologies and refer to OWL domain service ontologies (SOs).

The approach can be easily generalized to other lighter semantic annotation languages for Web services such as SAWSDL. Using OWL-S as the reference semantic description language, the large set of service descriptions given in the OWL-S Service Retrieval Test Collection 2.2. (OWLS-TC)4 can be considered available to the peer; referring to this set of services we stress that:
1) several services may be collected from the Web or from available repositories (OWLS-TC provides more than 1000 for the OWL-S 1.1 language),
2) several ontologies are referred to in the service descriptions (OWLS-TC provides 43 ontologies), and 3) these ontologies may concern different domains (OWLS-TC provides services from seven different domains).
The IR approach (aimed at locating relevant services related to the SQL query) requires a formal representation of the service descriptions stored in the repository, and it is based on full text indexing which extracts terms from six specific sections of a service description: service name, service description, input, output, pre-condition and post-condition. As an example, the service "City Hotel Service" from the OLWS-TC collection imports two domain ontologies (Travel and Portal) and is described by: the name "CityHotelInfoService", the text descriptions "This service returns information of a hotel of a given city", and the ontology concepts City (from the Portal ontology) and Hotel (from the Travel ontology) as input and output respectively. While the service name and description consist of short text sections, input and output refer to domain SOs, namely, a portal and a travel ontology

GSO construction
The Global Service Ontology (GSO) is built by considering the following two step procedure:
• loosely merging each service ontology O such that i in O for i in IO (e.g. the portal and travel ontologies in the reference example) – the process takes into account ontology imports recursively.
• associating a concept Ci with each i in IT , by introducing a class Terms subclass of Thing in the GSO and stating that for every i in IT , Ci is subclass of Terms (e.g. the term "reservation" occurring in the service name or service description sections).

With "loosely merging" we mean that SOs are merged asserting that their top concepts are all subclasses of Thing without attempting to integrate similar concepts across the different integrated ontologies. Therefore, if the source SOs are consistent, the GSO can be assumed to be consistent because no axioms establishing relationships among the concepts of the source SOs are introduced (e.g. axioms referring to the same intended concept would actually refer to two distinct concepts with two different URI).
Loose merging is clearly not the optimal choice with respect to ontology integration, but since the existent techniques do not allow to integrate ontologies in a completely automatic way, this is the only technique that guarantees consistency, without requiring a further user intervention. Moreover, since the XIRE retrieval component is based on IR techniques, approximate solutions to the ontology integration problem can be considered acceptable; instead, the whole GSO building process need to be fully automatized.

Construction of the GLSO and the Semantic Similarity Matrix
The GSO may result extremely large in size, which makes the semiautomatic process of mapping the GSO to the SPDO more expensive; moreover, only a subset of the terms of the ontologies used and recursively imported by the SWS descriptions are actually relevant for describing the SWS.
To solve this problem a technique to reduce the ontology size is exploited and a GLSO (Global Light Service Ontology) is obtained.
We provide an ontology module extraction algorithm that exploits a traversal-based approach. The algorithm, named SSiMap K-ex (Semantic Similarity and Mapping-driven K ontology module Extraction), is based on the extraction of (i) a subtree of the tree representing the unfolded concept hierarchy of the GSO, and (ii) a set of properties whose domains include the concepts in such a subtree. The subtree is built by traversing upwards the subclass relation until the root starting from the index terms I, and by traversing downwards the subclass relation through a path of an arbitrary length k.

Mapping of Data and Service Ontologies
Mappings between the elements from the SPDO and the GLSO are generated by exploiting the clustering algorithm used in MOMIS for creating the SPDO. In particular, the clustering algorithm relies on an input matrix where the columns (the rows) represent the source schema elements and the value in a cell weights

the relatedness of the schema elements that are in the corresponding row and column. To compute the mappings between the SPDO and the GLSO, the clustering algorithm requires an input matrix with the SPDO schema names and the names of the GLSO concepts. The weights are obtained by taking into account measures based on syntactic, lexical and structural similarity that are computed exploiting the knowledge about the sources in the MOMIS common thesaurus.
In particular, the syntactic similarity measures the similarity between the names used for describing the elements in the SPDO and in the GLSO. Several string similarity metrics have already been proposed [10], e.g., Jaccard, Hamming, Levenshtein, etc. As our approach is independent from the similarity metrics selected we leave this choice to the application.
String similarity, unfortunately, may fail in highly heterogeneous environments that lack a common vocabulary. In these cases, it is critically important to be able to capture the meaning of a word and not only its syntax. For this reason, we employ a lexical similarity measure based on WordNet that evaluates the terms on the basis of their synonyms, hypernyms and hyponyms. In particular, our approach builds a weighted network of relationships exploiting the terms in Wordnet and the relationships in the MOMIS Common Thesaurus. The weight of the path connecting two terms having minimum weight measures the lexical similarity of these terms.
Finally, a structural similarity measure compares the concept structures for evaluating their similarities. Concepts with similar structures have a huge structural similarity value.
The algorithm for generating the input matrix is highly customizable, thus making possible for the user to select the importance of each measure according to the source structures and contents. Moreover, the user may select the algorithm threshold in order to obtain big clusters (and consequently less selective associations between the SPDO and GLSO elements) or clusters where only strictly related elements are grouped. Mappings are then automatically generated exploiting the clustering result. The following cases are possible:

• A cluster contains only SPDO classes: it is not exploited for the mapping generation; this cluster is caused by the selection of a clustering threshold less selective than the one chosen in the SPDO creation process.
• A cluster contains only GLSO classes: it is not exploited for the mapping generation; it means that there are descriptions of Web Services which are strongly related.
• A cluster contains classes belonging both to the SPDO and the GLSO: this cluster produces for each SPDO class a mapping to each GLSO class. Mappings between the attributes of the classes are generated on the basis of the relationships held in the MOMIS Common Thesaurus.

As an example, consider the SPDO described in Figure 1, and a piece of the GLSO concerning the class Accomodation and the attributes this class is domain of; using a dotted notation in the form of "concept.property" this piece of ontology is represented as follows:

Accomodation
Accomodation.Denomination
Accomodation.Location
Accomodation.Country

The following mappings are generated with the application of our technology:

Hotel --> Accomodation
Hotel.Name --> Accomodation.Denomination
Hotel.City --> Accomodation.Location
Hotel.Country --> Accomodation.Country

Observe that clusters can be exploited to correct false dissimilarities in the SSM. However, clusters provide also another solution to the false dissimilarities problem: when two terms t1 and t2 of the GLSO are clustered together, they are mapped to the same term s of the SPDO; when a query formulated in the SPDO terminology contains the term s, both t1 and t2 will be extracted as keywords and used to retrieve relevant services.

Data and eService Retrieval - Execution Time

Let us introduce a query expressed in the SQL language:
select <select_attribute_list>
from <from_class_list>
where <condition>

The answer to this query is a data set from the data sources together with a set of ranked services which are potentially useful, since they are related to the concepts appearing in the query and then to the retrieved data.
The query processing is thus divided into two steps:
• a data set from the data sources is obtained with a SQL query processing on the integrated SPDO view
• a set of services related to the query is obtained by exploiting the mapping between SPDO and GLSOs and the concept of relevant service mapping, by executing the IR engine.

Data results are obtained by exploiting the MOMIS Query Manager which rewrites the global query as an equivalent set of queries expressed on the local schemata (local queries); this query translation is carried out by considering the mapping between the SPDO and the local schemata. Since MOMIS follows a GAV approach, the query translation is performed by means of query unfolding. Results from the local sources are then merged by exploiting the reconciliation techniques.
As the query processing on an integrated view is already well described in the literature, in the following we focus our attention on the queries for services. Services are retrieved by the XIRE (eXtended Information Retrieval Engine) component, which is a service search engine based on the vector space model, and implemented with the open source libraries Lucene; the query is a set of weighted terms.
The process for the creation of the set of weighted terms is represented in Figure 3. The process starts by selecting the terms in the SQL query that appear in the SPDO. Then, the query manager checks which elements of the GLSO are mapped on these terms of the SPDO; these keywords are expanded on the basis of the SSM, and a set of weighted term (the query vector) is provided to the information retrieval engine which will retrieve the related services.

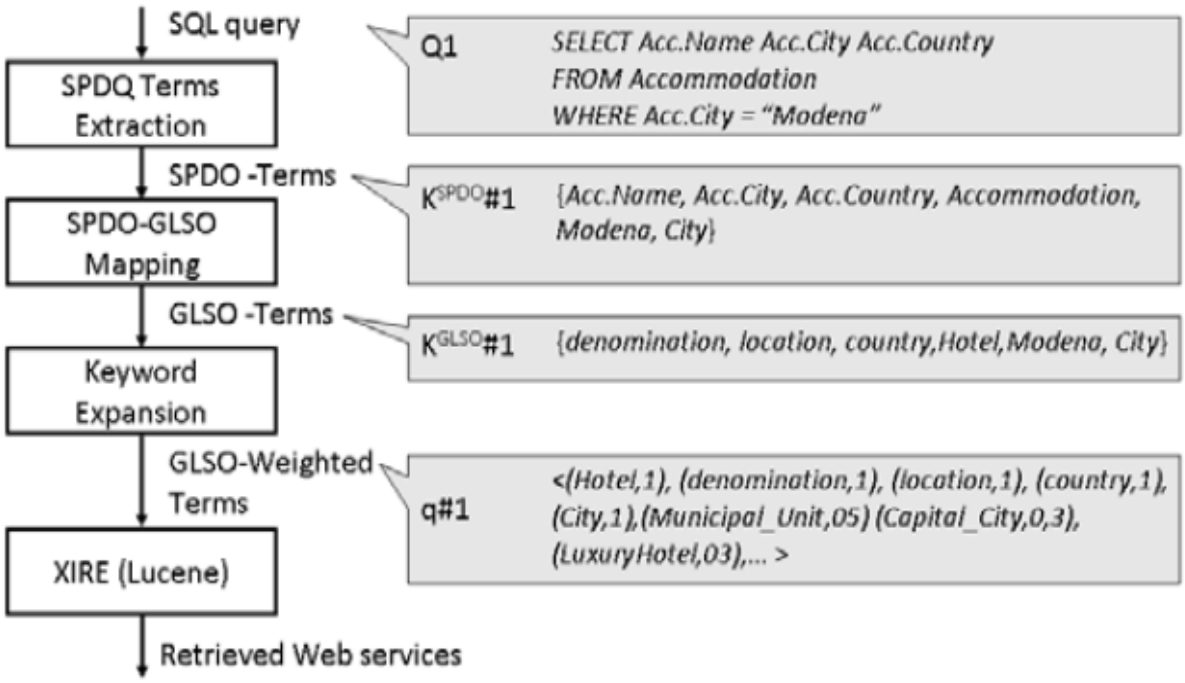

Figure 3: The query processing steps and their application to the reference example

eService Retrieval

Terms extraction. Given a SQL query expressed in the SPDO terminology, the set of KSPDO of terms extracted from an SQL query consists of: all the classes given in the "FROM" clause, all the attributes and the values used in the "SELECT" and "WHERE" clauses, and all their ranges defined by ontology classes. As an example, the set of terms extracted from the query Q1, consists of the set KSPDO#1 represented in Figure 3.

Terms rewriting The set of terms KSPDO extracted from the user query are rewritten in a set of keywords KGLSO exploiting the mappings between the SPDO and the GLSO. Let us define a data to service ontology

mapping function μ : SigSPDO $\rightarrow$ P(SigGLSO). The function, given a term s $\in$ SPDO returns a set of

keywords T $\subseteq$ SigGLSO iff every t $\in$ T is in the same cluster of s. Given a set of terms KSPDO = {k0, ..., km},

each term ki with $0 \leq i \leq m$ is replaced by the set of keywords returned by μ (k0). By using the mappings described above, and assuming μ (Modena) = Modena, and μ (City) = City, the set of keywords obtained in the reference example is the set KGLSO#1 represented in Figure 3.

Keywords expansion Semantic similarity between GLSO terms defined in the SSM is exploited to expand

the KGLSO set with additional terms into a vector q =< w1, ..., , wn >, where for $1 \leq i \leq n$, ki $\in$ SigGLSO, and

wi are weights that represent the importance of each keyword in describing the interesting services. The vector q is obtained by associating each keyword with a weight equal to 1, and adding a set of terms that in the SSM are similar to the given keywords up to a given threshold weight according to their similarity w.r.t. the given keywords.

More formally, let simsets(t) $\subseteq$ SigGLSO be the set of terms of the GLSO such that their similarity w.r.t. t is

greater than a given threshold th. Given a set of keywords KGLSO = {k0, ..., km}, the vector q is obtained as

follows: all the keywords ki $\in$ KGLSO, with $1 \leq i \leq m$, are inserted in q and are associated with a weight wi =

1; for every ki $\in$ KGLSO the set simsets (ki) is inserted in q, and each element e $\in$ simsets (ki) is associated

with a weight we = sim (ki, e); duplicate terms in q are discarded, keeping the terms associated with the greatest weight.
For example, let us consider the set of keywords KGLSO#1 given in the reference example. Assume to set

the normalized similarity threshold th = 0, 95; simset0,95 (City) $\subseteq$ {Municipal Unit,Capital City}, sim

(City,Municipal Unit) = 0, 999 (City is subclass of Municipal Unit in the Travel ontology) and sim (City,Capital City) = 0, 995 (Capital City is subclass of City); a piece of the resulting weighted term query vector q#1, including Municipal Unit,Capital City and LuxuryHotel (added in an analogous way based on the ontology including Hotel), is represented in Figure 3.

Services retrieval. Query evaluation is based on the vector space model; by this model both documents (that is Web Service descriptions) and queries (extracted queries) are represented as vectors in a n-dimensional space (where n is the total number of index terms extracted from the document collections). Each vector represents a document, and it will have weights different from zero for those keywords which are indexes for that description. The value of such weight is computed according to the weights of the six sections of the service description in which the keyword appears. We assume that the implicit constraint specified in a user query, when selecting a query term (a single keyword) is that it must appear in at least one section of a service description in order to retrieve that service. Based on the above assumptions the weight which at query evaluation time is associated with a keyword and a service description is equal to the maximum of the weights of the service sections in which the keyword appears.

System Architecture

To evaluate our approach we extended the existing data integration system MOMIS and we coupled it with XIRE (eXtended Information Retrieval Engine) an intensive semanticWeb service retriever we developed. The global architecture of our prototype is shown in Figure 4.

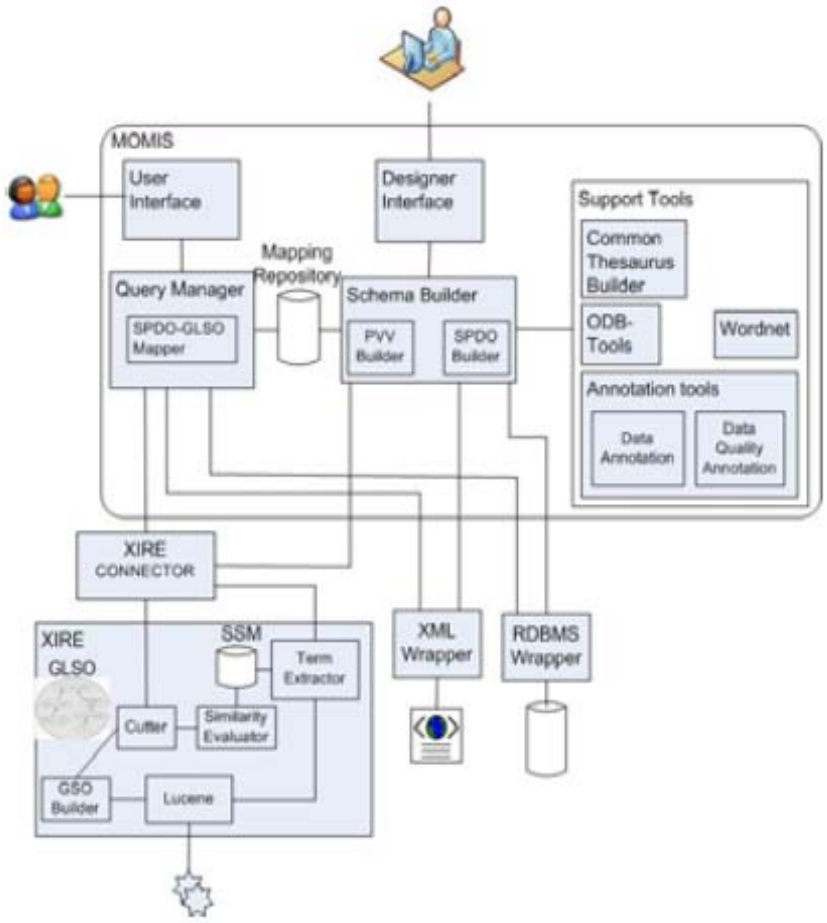

Figure 4: Prototype architecture

MOMIS

MOMIS is a suite of software modules implementing the integration process. Its main components are:

• Schema builder which is in charge of generating the global virtual view of the data sources and the PVV that represents both the data sources and the services available in a peer.

• Support tools, which are a set of components aiming at enriching the descriptions of the sources and services with metadata exploited by the schema builder;

• Query manager, which is in charge of evaluating a user query solving it with respect to the data sources and the available services;

• Wrappers that are software modules with the role of managing the interactions with the data sources.

The Schema Builder

The Schema builder is mainly composed of two modules, one in charge of creating a global virtual view of data sources (i.e., the SPDO builder) and the second one computing the mappings from the global view of data into the concepts that web services refer to (i.e., the PVV builder).

The SPDO builder implements a hierarchical clustering tool that groups the descriptions of the data sources. This component interacts with the wrappers and the Support Tools to obtain the data source descriptions and the metadata in the Common Thesaurus representing the relationships among the data sources. The result of the process is a SPDO, which is composed of a set of Global Classes, each one with a Mapping Table showing the local data source elements represented by the Global Class.

The PVV builder is a component that provides the mappings between the SPDO and the GLSO (i.e. the ontology representing the web services in a peer) elements. The PVV builder implements a clustering algorithm that works on the basis of the descriptions of the GLSO elements (extracted by a specific wrapper), the PVV description and a set of semantic relationships computed by the Support Tools component. The mappings resulting from this process express one to one matches of SPDO classes and attributes into GLSO concepts and properties.

The Support tools

This components contains some important modules exploited by the Schema Builder and the Query Manager at run time. In particular, the Annotation tools are modules that associate for each term in the database (table and attribute names) some metadata describing its lexical meaning with respect to a lexical reference (in our case WordNet) and some quality measures about the data (in particular the accuracy, the completeness and the currency).

Annotations are exploited by the Schema Builder for finding the similar descriptions of real world objects in different sources and by the Query Manager for selecting the most promising source when more than one source contain data about the same domain. The Common Thesaurus Builder is the component in charge of creating and managing a set of inter and intra source relationships between the available sources. Some relationships are built by exploiting the annotations, some other relationships are computed by exploiting description logics techniques implemented in the ODB-Tool component.

The Query manager

The MOMIS Query Manager is the component in charge of solving a user query, i.e. executing the query over the SPDO and extracting some relevant keywords to be exploited by XIRE for retrieving the related web services.

Concerning the query execution on the data sources, when the component receives a query, it rewrites the global query as an equivalent set of queries expressed on the local schemata (local queries); this query translation is carried out by considering the mappings between the SPDO and the local schemata. Since MOMIS follows a Global as View (GAV) approach, where the contents of the mediated schema is expressed in terms of the local sources schemata, this mapping is expressed by specifying, for each global class C, a mapping query QC over the schemata of the local classes belonging to C. The system automatically generates the mapping query QC, by extending the Full Disjunction (FD) operator and exploiting the Data Transformation Functions, which are defined by the user and represent the mapping of local attributes into the SPDO attributes. The query translation is thus performed by means of query unfolding, i.e. by expanding a global query on a global class C of the SPDO according to the definition of the mapping query QC. Results from the local sources are then merged exploiting reconciliation techniques proposed to the user.

The relevant keywords extracted from a query are the one identifying schema elements and searched values. By means of the mappings computed by the PVV builder, for each keyword, the correspondent terms in the GLSO are extracted, if exists. Such terms are then sent to XIRE by means of a specific wrapper.

The Wrappers

The MOMIS wrappers translate the source data structures into ODLI3. Their role is to deal with the diversity of the data sources thus allowing MOMIS to pay no attention to the language details of the different data sources representing every type of data source in the same language. Wrappers are available for different kind of data sources, ranging from different database management systems to semi-structured data like XML, RDF, OWL formats.

Wrappers logically guarantee two main operations:

• getschema() translates the schema from the original format into ODLI3, dealing with the necessary data type conversions

• runquery() executes a query on the local source. The MOMIS Query Manager translates a query on the SPDO (a global query) into a set of local queries to be locally executed by means of wrappers.

The XIRE connector

The XIRE connector is the component in charge of managing the interactions between the MOMIS and the XIRE systems. The connector mainly supports two tasks:

• getGLSOschema() translates the schema of the GLSO (expressed in OWL) into ODLI3.

• serviceDiscovery() enables the discovery of the services on the basis of a query on the SPDO. The MOMIS Query Manager translates a query on the SPDO (a global query) into a set of keywords that will be exploited by XIRE to get a list of web services relevant to the global query.

XIRE

The main modules of XIRE are the following:

• Lucene is the core of the module and is a information retrieval tool developed by Apache Software Foundation, available at http://lucene.apache.org/.

• GSO Builder which is in charge of building the GSO based on the indexes built by XIRE.

• Cutter is the module realizing the module extraction algorithm.

• Similarity Evaluator which is the components devoted to the definition of the semantic similarity matrix.

• Term Extractor, which, at query time, is in charge of extracting the set of weighted keywords provided to Lucene from the terms of the GLSO.

GSO Builder

The GSO Builder is in charge of building the global service ontology starting from the list of terms extracted by the semantic web service descriptions provided by the Lucene component. For each term, that is also a reference to an ontological concept, the GSO Builder imports the whole ontology associated to it. In such a way all related concepts are added to the GSO. GSO Builder makes use of the Jena framework, in particular the OWL API.

Cutter

Cutter is the module in charge of lightening, at set-up time, the GSO for better handling by implementing the SSiMap K-ex algorithm; the configuration used set K = 2.

Similarity Evaluator

At set-up time Similarity Evaluator builds a similarity matrix (SSM - see Figure 4) based on GLSO concepts: the matrix contains the similarity value for each couple of GLSO concepts. The Java library SimPack has been used for building the Similarity Matrix. SimPack provides algorithms to calculate many syntactic and semantic similarity metrics (e.g. Conceptual Similarity, Lin, Resnik, ScaledShortestPath and ShortestPath), which can be used, and eventually combined, in the XIRE component by configuring the Similarity Evaluator. To provide the normalized similarity value we exploited the Common Distance Conversion available in the SimPack library.

Term Extractor

Term extractor exploits the similarity matrix (SSM - see Figure 4) by finding, for each keyword provided by the XIRE connector the most similar concepts. This is realized by using a normalized similarity threshold (in a range from 0 to 1), that, for the experiments is set to 0.95, in order to restrict as much as possible the terms proliferation.

Referenze

[1] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M. Vincini, "A Semantic Approach to Information Integration: the MOMIS project", Sesto Convegno della Associazione Italiana per l'Intelligenza Artificiale, AI*IA 98, Padova IT, Settembre 1998.

[2] D. Calvanese, S. Castano, F. Guerra, D. Lembo, M. Melchiori, G. Terracina, D. Ursino, M. Vincini: "Towards a comprehensive methodological framework for integration", 8th International Workshop on Knowledge Representation meets Databases (KRDB-2001)

[3] D. Beneventano, S. Bergamaschi, Claudio Sartori, Maurizio Vincini, "ODB-Tools: a description logics based tool for schema validation and semantic query optimization in Object Oriented Databases", Proceeding of the Fifth Conference of the Italian Association for Artificial Intelligence (AI*IA97): Advance in Artificial Intelligence, LNAI Vol. 1321, Springer. Berlin, September 17-19, 1997, ISBN 3-540-63576-9.

[4] D. Beneventano, S. Bergamaschi, J. Gelati, F. Guerra, M. Vincini: "MIKS: an agent framework supporting information access and integration", Intelligent Information Agents - The AgentLink Perspective, (editor S. Bergamaschi, M. Klusch, P. Edwards, P. Petta) - March 2003, Lecture Notes in Computer Science N. 2586 - Springer Verlag, ISBN 3-540-00759-8.

[5] S. Bergamaschi, F. Guerra, M. Vincini: "A peer-to-peer information system for the semantic web", in proceedings of the International Workshop on Agents and Peer-to-Peer Computing (AP2PC03), held in AAMAS 2003 International Conference on Autonomous Agents and MultiAgent Systems Melbourne, Australia, July 14, 2003

[6] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini "ODB-QOptimizer: a tool for semantic query optimization in OODB." Int. Conference on Data Engineering ICDE97, Birmingham, UK, April 1997.

[7] S.Bergamaschi, P. Bouquet, D. Giacomuzzi, F. Guerra, L. Po, and M. Vincini: MELIS: an incremental method for the lexical annotation of domain ontologies, in International Journal on Semantic Web and Information Systems (IJSWIS) 3(3), p.p.57-80 2007