BUILDING A TOURISM INFORMATION PROVIDER WITH THE MOMIS SYSTEM

S. Bergamaschi[1,2], D. Beneventano[1,2], F.Guerra[1], M.Vincini[1]

[1] Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia

[2] IEIIT-CNR Istituto di Elettronica e di Ingegneria dell'Informazione e
delle Telecomunicazioni – Bologna

Abstract
The tourism industry is a good candidate for taking up Semantic Web technology. In fact, there are many portals and web sites belonging to the tourism domain which promote tourist products (places to visit, food to eat, museums, …) and tourist services (hotels, events, …), published by several operators (tourist promoter associations, public agencies,…).
This paper presents how the MOMIS system may be used for building a Tourism Information provider by exploiting the tourism information that is available in internet websites. MOMIS, (Mediator envirOnment for Multiple Information Sources), is a mediator framework that performs information extraction and integration from heterogeneous distributed data sources and includes query management facilities to transparently support queries posed to the integrated data sources.
Keywords: Ontologies, Integration, Mediator-based systems, Intelligent Integration of Information

1. INTRODUCTION

The large amount of web sites makes available a lot of information concerning technical, cultural and entertainment topics. The Internet is the place where it is possible to find research projects published by universities, information about cities, museums, exhibitions published by town halls and specific agencies, information about products and dealers. By means of the Internet, a user can choose and buy a product or a service, directly from the producer, without any intermediation.

Looking at the tourism domain we can see many portals and web sites promoting tourist products (places to stay, food to eat, museums, …) and tourist services (hotels, events, …) available; several operators (tourist promoter associations, public agencies,…) publish relative information. However, the information about a place or an event is frequently widespread in different specific websites, due to the specialization of the information publisher. Thus, if a user wants to have a “global” knowledge about a location, he has to navigate through several websites. This issue generates multiple researches on search engines, and the possibility of having incomplete or old outdated information is high.

Two kinds of users are mainly interested in this information: tourist promoters and frequent travellers. Tourist promoters are interested in collecting information from different sources in order to create a complete offer for their customers. Travellers aim at personally organizing their travels and looking for destinations, hotels, ...

Both types of users need easy-to-retrieve, complete and up-to-date information. Information has to be complete: the data coming from several web sites (e.g. accommodation and events/local attractions) has to be merged into a unique answer including all the results. Moreover, the information has to be correct: old information could be wrong and consequently create some false expectations for visitors.
Thus, it is possible to summarize the main desiderata for a tourist information system into the definition of methods for:

a) extracting and fusing the information coming from different (and heterogeneous) information sources (e.g. web sites, and tourist information providers), and
b) guaranteeing that the information is up-to-date.

The paper presents the MOMIS system (Mediator envirOnment for Multiple Information Sources) (Bergamaschi01), a mediator-based system that extracts and integrates information from heterogeneous distributed data sources and with query management facilities to transparently support queries posed to integrated data sources. The MOMIS framework consists of a language and two main components:

- The ODL-I3 language extends an object-oriented language, with an underlying Description Logic; it is derived from the standard ODL-ODMG (Cattell00).
- The Ontology Builder: the various sources integration is performed in a semi-automatic way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL-I3 descriptions of source schemas with a combination of clustering techniques and Description Logics. This integration process gives rise to a virtual integrated view of the underlying sources (the Global Schema, GVV) for which mapping rules and integrity constraints are specified to handle heterogeneity.
- The MOMIS Query Manager is a coordinated set of functions which takes an incoming query, decomposes the query according to the mapping of the GVV on the local data sources relevant to the query, sends the subqueries to these data sources, collects their answers, performs any residual filtering necessary, and finally delivers the answer to the user.

The MOMIS system is based on a conventional wrapper/mediator architecture, and provides methods and open tools for data management in Internet-based information systems (Figure 1). MOMIS development begun as a joint collaboration between the University of Modena and Reggio Emilia and the University of Milano and Brescia, within the INTERDATA national research project. Research activities continue within the SEWASIE European research project (IST-2001-34825) (www.sewasie.org)

MOMIS can analyze the contents of several tourist related web sites and build a Global View. This Global View (GVV) is “virtual” i.e. it is not materialized: data resides in “local” sources and the view is an entry point for data retrieving. Consequently, only the changes in information source structures have a side effect on a built GVV. Data changes do not have any influence on it. Moreover the existence of semantic tags in the sources, or the semantic annotations with respect to a lexical ontology (e.g. WordNet) are used to build the GVV which can be seen as a domain ontology of the involved sources. The GVV can be exported in RDFS and OWL thus guaranteeing interoperability with other external applications/ontologies or external users.

In this paper, we show how the MOMIS system can be used for building a Tourist Information system for a Tourist Provider. In particular, we describe a view in the tourist domain based on the restaurants industry. We integrate three different web sites including the most important Italian restaurant guides. The GVV obtained is part of a larger knowledge base where restaurant descriptions are correlated to places and places are correlated to events in those places. In the integration process, we focus on the following:

1) How we are able to recognize the same restaurant retrieved from different data sources;

2) How we can select the most up-to-date information among different sources.

Then, we present some queries and show the answers generated by MOMIS, which are obtained by fusing the information coming from the data sources.

The outline of the paper is as follows: the first section describes the use of MOMIS to create a tourist information system: in particular Section 2 describes the web sites integration approach, section 3 sketches out the querying process. Section 4 shows the evolution of MOMIS architecture in web services to improve its interoperability with other systems. Finally Section 5 compares MOMIS with other system and Section 6 gives conclusions.

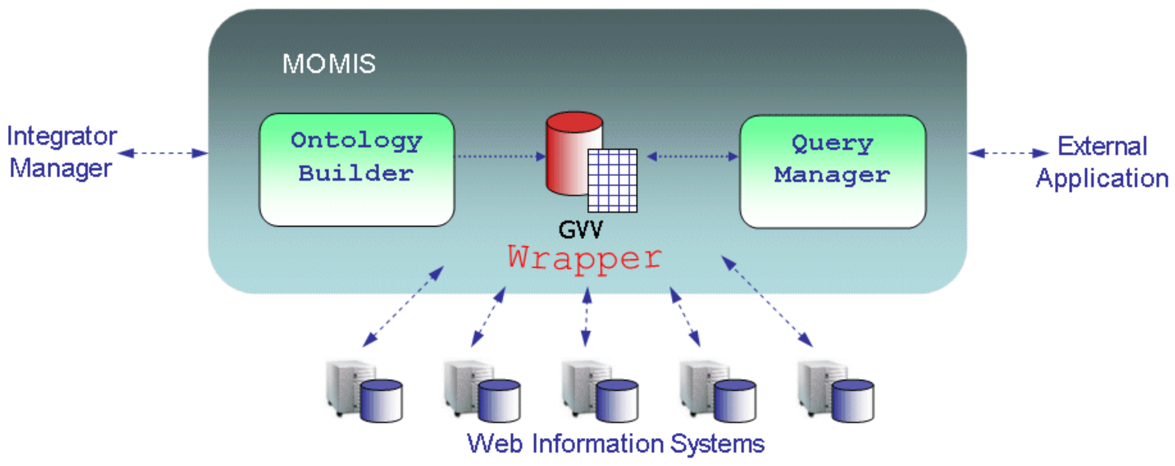


Figure 1. The MOMIS Architecture

2. THE MOMIS INTEGRATION METHODOLOGY

In this section, we describe the information integration process for building the GVV. The process, shown in Figure 2, gives rise to an integrated information system relating to several specific information sources. The GVV-generation process 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 ODL$_{I3}$.

2) Local source annotation with WordNet. The integration designer chooses a meaning for each element of a local source schema, according to the WordNet lexical ontology (hhtp://www.cogsci.princeton.edu/~wn). This phase may be executed semi-automatically: a tool helps the integration designer make the right the choice by proposing a WordNet concept for each source element.

3) Common thesaurus generation. Starting from the annotated local schema, MOMIS constructs a set of relationships describing inter- and intraschema knowledge about classes and attributes of the source schemata..

4) GVV generation. The MOMIS methodology, applied to the common thesaurus and the local schemata descriptions, generates a global schema and sets of mappings with local schemata.

5) GVV annotation. Exploiting the annotated local schemata and the mappings between local and global schemata, the MOMIS system semi-automatically assigns a meaning to each element of the global schema.

The above five phases are described in the following sections. The Ontology Builder Tool helps the integration designer in all the GVV generation process phases. In particular, we introduce the MOMIS approach by creating a knowledge base about Italian restaurants. As previously described this knowledge base is part of a bigger one containing more tourist information about Italian places, events and restaurants.

2.1 The ODLI3 Language

As a common data model for integrating a given set of local information sources, MOMIS uses an object-oriented language called ODL*I3*, which is an evolution of the OODBMS standard language ODL. ODL*I3* extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas: SYN (synonym of), BT (broader terms), NT (narrower terms) and RT (related terms). By using ODL*I3*, only one language is used to describe both the sources (the input of the synthesis process) and the GVV (the result of the process). The translation of ODL*I3* descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL*I3* concept corresponds to a *Class* of the Semantic Web standards, and ODL*I3* relationships are translated into *properties*.

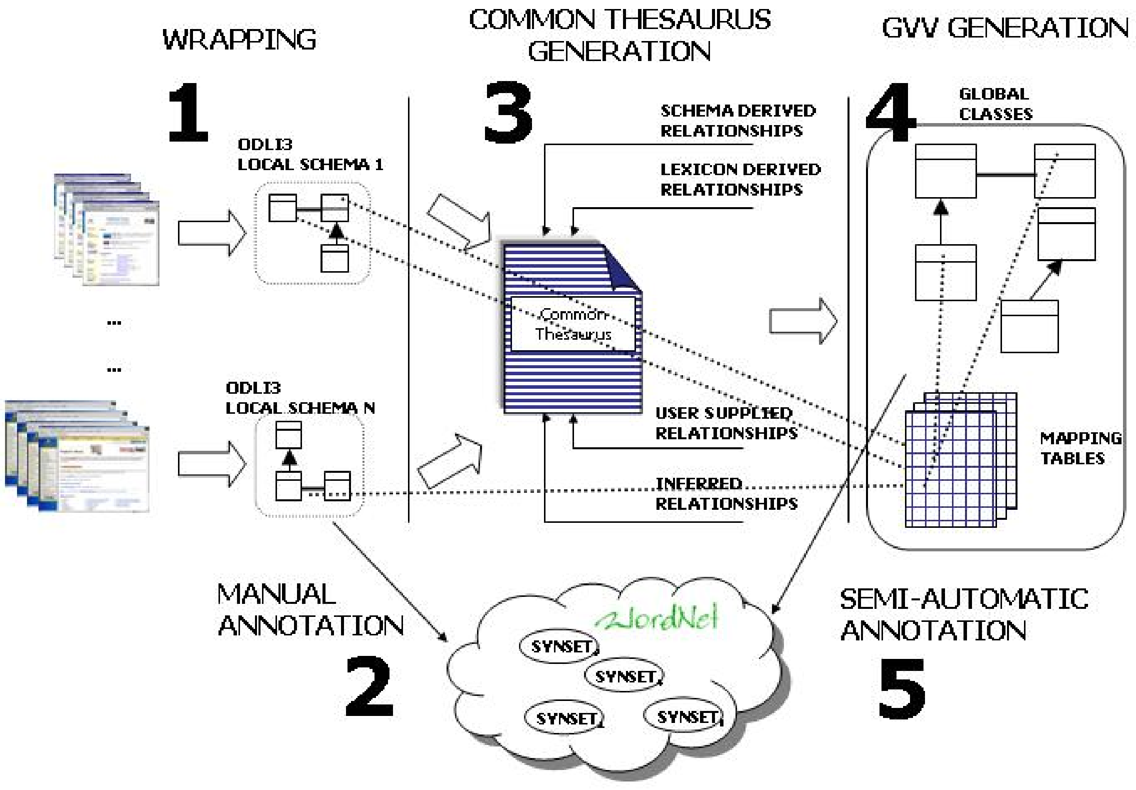


Figure 2. Integration process overview.

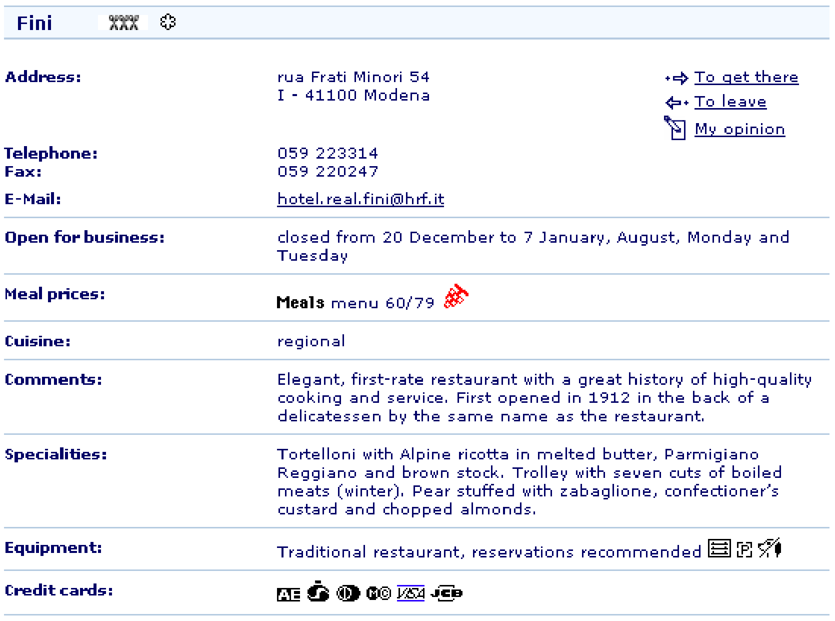
Figure 2 shows the local schemas' generation, where local schemas are annotated according to the lexical ontology WordNet, the Common Thesaurus generation, and finally the GVV global classes. In particular, GVV global classes are connected by means of mapping tables to the local schemas and are (semi-automatically) annotated according to WordNet.

2.2 Wrapping: extracting data structure for sources

A wrapper logically converts the source data structure into an ODL*I3* schema. The wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources. For conventional structured information sources (e.g. relational databases), schema description is always available and can be directly translated. For web content, this is mainly available in the form of HTML documents: such documents do not separate data from presentation and are ill-suited for being the target of database queries and most other forms of automatic processing. This problem has been addressed by much work on so-called Web wrappers, programs that extract the relevant information from HTML documents and translate it into a more machine-friendly format such as XML. The wrapping problem has been addressed by a substantial amount of work (see e.g. TSIMMIS (Li98), FLORID (Ludascher, 98), DEByE (Laender, 02), W4F (Sahuguet, 01), XWrap (Liu, 00), and Lixto (Gottlob, 04) for some research systems): in our approach we used Lixto to translate the website data in XML format, then we developed an XML wrapper that generates the related XML Schema (an XSD file) and loads the XML data into a relational database. Our XML wrapper is based on MS .net framework and automatically updates the data extracted from the web by means of script daemons into the relational database.

Italian restaurant source wrapping. The first phase of the integration process is choosing of the significant information sources and their translation into a machine readable/processable format. On the web there are several guides about Italian restaurants that are published, in particular the

“Michelin” and the “Veronelli” guide which are freely available. We also integrated a restaurant from the www.acena.it site. This site in particular is considered to be the first official guide on restaurants meant for Internet and its reliability is guaranteed by the quality of the sources, fed by the well known newspaper guides on restaurants such as "L'Espresso", "Gambero Rosso", the "Accademia Italiana di Cucina", “Ristoranti d’Europa” edited by Sperling & Kupfer, respecting international copyrights. In this paper, we show how to create a tourist integrated system by collecting the information at the www.viamichelin.com, www.veronelli.com, www.acena.it web sites. We used Lixto software in order to wrap the web pages: Figure 3 shows a page extracted from the www.viamichelin.com site concerning a Modena restaurant (the “Fini” restaurant)



Fini

Address:	rua Frati Minori 54 I - 41100 Modena	To get there To leave My opinion
Telephone: **Fax:**	059 223314 059 220247	
E-Mail:	hotel.real.fini@hrf.it	
Open for business:	closed from 20 December to 7 January, August, Monday and Tuesday	
Meal prices:	**Meals** menu 60/79	
Cuisine:	regional	
Comments:	Elegant, first-rate restaurant with a great history of high-quality cooking and service. First opened in 1912 in the back of a delicatessen by the same name as the restaurant.	
Specialities:	Tortelloni with Alpine ricotta in melted butter, Parmigiano Reggiano and brown stock. Trolley with seven cuts of boiled meats (winter). Pear stuffed with zabaglione, confectioner's custard and chopped almonds.	
Equipment:	Traditional restaurant, reservations recommended	
Credit cards:	AE	

Figure 3. A www.viamichelin.com web page about an Italian restaurant

Lixto allows you to find and select interesting information from each HTML page and then create a unique file with all the information. The file created is an XML file and in our example contains all the information about the Italian restaurant stored in the web site.

Lixto is a visual wrapper generator, i.e. the process of interactively defining a wrapper from one (or a few) example document(s) using mainly “mouse clicks" is supported by a strong and intuitive design metaphor. During this visual process, the wrapper program should be automatically generated and should not actually require a human designer to know the wrapper programming language.

The wrapping phase extracts all possible data for each interesting field previously selected assigning a label to each one. This label describes the kind of data that is wrapped: e.g. Cuisine, Specialties, Equipment, Credit cards in Figure 3.

Figure 4 shows a piece of the XML file we obtained by applying Lixto to the page in Figure 3.

```
   <Restaurant>
    <Name>Fini</Name>
    <Category>3</Category>
    <Flower>1</Flower>
    <Address>rua Frati Minori 54</Address>
    <ZipCode>41100</ZipCode>
    <City>Modena</City>
    <Province>MO</Province>
    <Telephone>059 223314</Telephone>
    <Fax>059 220247</Fax>
    <Email>hotel.real.fini@hrf.it</Email>
    <ClosingDay>closed from 20 December to 7 January, August, Monday and
Tuesday</ClosingDay>
    <Cousine>regional</Cousine>
    <Price>Meals menu 60/79 </Price>
    <Comment>Elegant, first-rate restaurant with a great history of high-quality cooking and service. First opened in 1912 in the back of a delicatessen by the same name as the restaurant.</Comment>
    <Specialities>Tortelloni with Alpine ricotta in melted butter, Parmigiano Reggiano and brown stock. Trolley with seven cuts of boiled meats (winter). Pear stuffed with zabaglione, confectioner’s custard and chopped almonds.</Specialities>
    <Equipment>Traditional restaurant, reservations recommended </Equipment>
    <CreditCard>AE, SI, Diners Club, Mastercard, VISA, JCB</CreditCard>
  <WebSource></WebSource>
  </Restaurant>
```

Figure 4. Part of the Lixto XML file obtained wrapping the page of fig. 3.

The available tools for querying XML files are still not so efficient and reliable as the DBMS is. Thus we designed a tool to turn XML files into relational databases. the XML2Relational module In particular generates a relational database corresponding to the XML file. The tool, developed by using and extending the Microsoft .Net framework associates an XML schema to the XML file in order to create a relational schema with the same semantics. Data is then stored in the database. Figure 5 shows a fragment of the XML Schema generated by wrapping the restaurants at www.viamichelin.com

```
  <?xml version="1.0" encoding="utf-8"?>
  <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="Restaurant">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="Name" type="xs:string" minOccurs="0"/>
          <xs:element name="Category" type="xs:string" minOccurs="0" />
          <xs:element name="Flower" type="xs:string" minOccurs="0/>
          <xs:element name="Address" type="xs:string" minOccurs="0/>
          <xs:element name="ZipCode" type="xs:integer" minOccurs="0/>
          <xs:element name="City" type="xs:string" minOccurs="0/>
          <xs:element name="Province" type="xs:string" minOccurs="0" msdata:Ordinal="6" />
  …
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NewDataSet" msdata:IsDataSet="true">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element ref="Restaurant" />
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

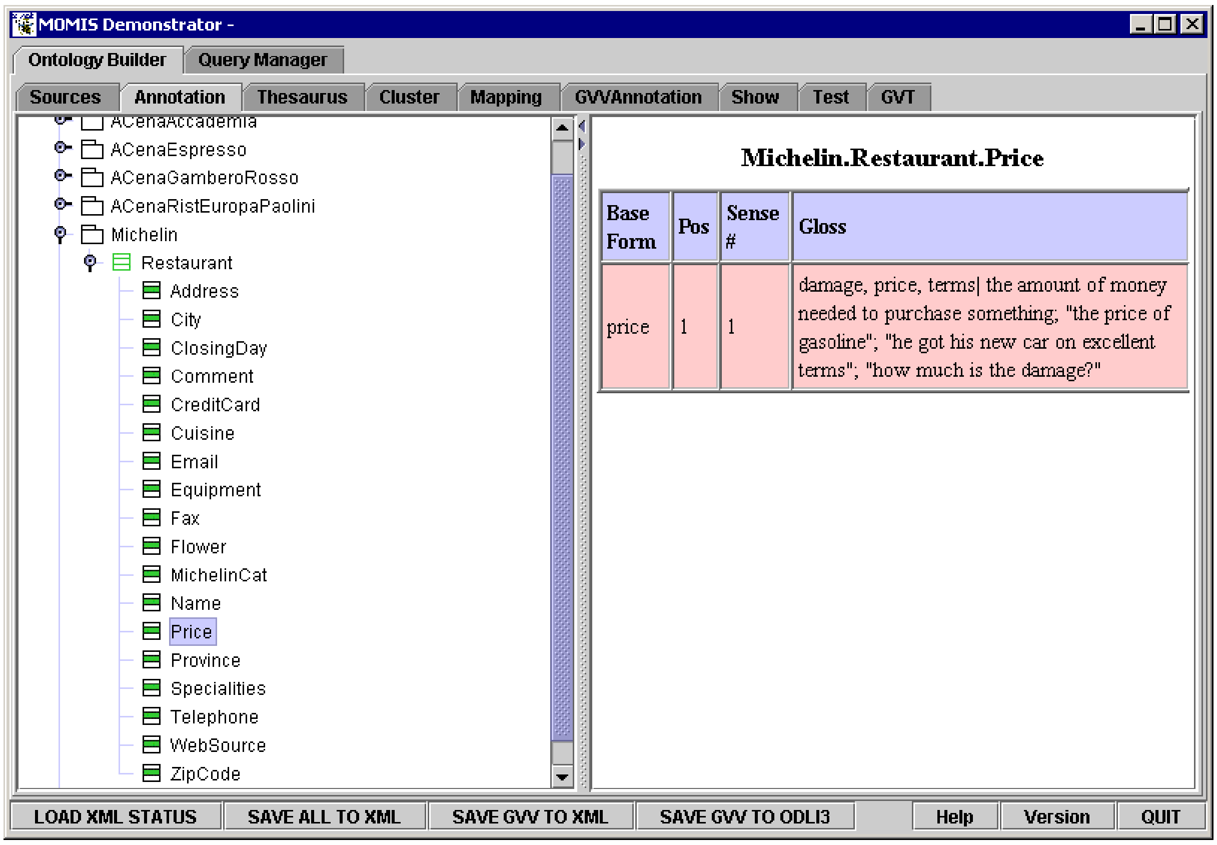
Figure 5. The XML Schema file representing the restaurants at www.viamichelin.it

In order to build the GVV, the same process has to be repeated for each source. the www.acena.it web site In particular contains three different guides. For each guide ("L'Espresso", "Gambero Rosso", the "Accademia Italiana di Cucina", Sperling & Kupfer and "Relais Chateaux"), the process has to be reiterated.

Finally, once the user has customized the site wrapping with Lixto, the wrapping operations may be automatically scheduled and executed in order to always keep the data stored in the database updated.

2.3 Semi-automatic annotation of a local source with WordNet

The goal of the annotation phase is to assign a name, and a set of meanings used by the WordNet (Miller, 95) lexical system to each local class and attribute of the local schema. The system automatically suggests a word form corresponding to the given term (if it exists) for each element of the local schema: the designer may confirm or change the word form or meaning of each element. When the annotation is related to a specific domain, our system allows the user to extend the WordNet ontology by adding new concepts and relating them to the native elements of WordNet (Benessi, 04).



Michelin.Restaurant.Price

Base Form	Pos	Sense #	Gloss
price	1	1	damage, price, terms| the amount of money needed to purchase something; "the price of gasoline"; "he got his new car on excellent terms"; "how much is the damage?"

Figure 6. The automatic annotation for the price attribute

For example, in Figure 6 the automatic annotation obtained by the system of the price attribute for the Michelin Restaurants is shown.

2.4 Common Thesaurus Generation

MOMIS constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of SYN, BT, NT, and RT relationships. The Common Thesaurus is constructed through an incremental process to which the following relationships are added:

schema-derived relationships: relationships holding at intra-schema level are automatically extracted by analyzing each schema separately. For example, by analyzing XML data files, BT/NT relationships are generated from pairs of IDs/IDREFs and RT relationships from nested elements.

lexicon-derived relationship: we use the annotation phase in order to translate relationships existing at the lexical level into relationships to be added to the Common Thesaurus. For example, the hypernymy lexical relation is translated into a BT relationship.

designer-supplied relationships: new relationships can be inserted by the designer, to capture specific domain knowledge. If a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema.

inferred relationships: Description Logics (DL) techniques of ODB-Tools (Beneventano, 97), (Beneventano, 98), (Beneventano03) are used to infer new relationships, by means of subsumption computation applied to a “virtual schema” obtained by interpreting BT/NT as subclass relationships and RT as domain attributes.

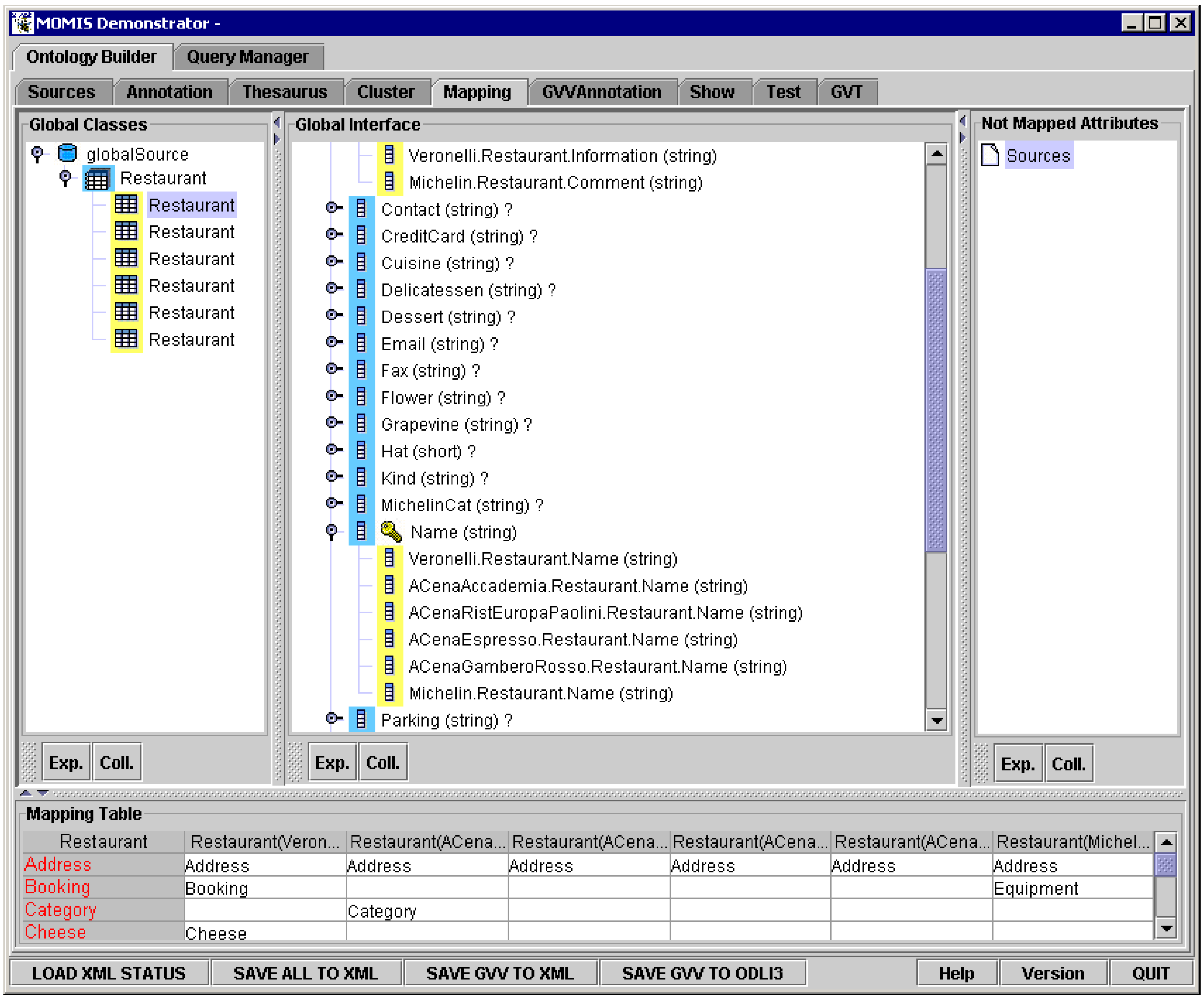
A detailed presentation of the methodology can be read in (Bergamaschi, 01), (Beneventano, 03b).

2.5 Global Virtual View (GVV) Generation

The MOMIS methodology allows us to identify similar ODL*I3* classes, that is, classes that describe the same or semantically related concept in different sources. Then, *affinity coefficients* are evaluated for all possible pairs of ODL*I3* classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name Affinity* coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients. Global affinity coefficients are then used by a hierarchical clustering algorithm, to classify ODL*I3* classes according to their degree of affinity (Castano, 01).

For each cluster Cl, a Global Class GC, with a set of Global Attributes (are defined for example) GA_1, …, GA_N , and a Mapping Table MT, expressing mappings between local and global attributes, . The Mapping Table is a table whose columns represent the local classes, which belong to the Global Class and whose rows represent the global attributes. An element `MT[GA][LC]` represents the set of local attributes of LC which are mapped onto the global attribute GA. How local attributes are mapped onto the global attribute GA is defined by means of Resolution Functions, which will be introduced in Section 3.3.3.

As an example, Figure 7 shows the Mapping Table for the Restaurant Global Class, where the global attribute “Name” has a counterpart for each web site.



Global Classes:

```
globalSource
  Restaurant
    Restaurant
    Restaurant
    Restaurant
    Restaurant
    Restaurant
    Restaurant
```

Global Interface:

```
    Veronelli.Restaurant.Information (string)
    Michelin.Restaurant.Comment (string)
Contact (string) ?
CreditCard (string) ?
Cuisine (string) ?
Delicatessen (string) ?
Dessert (string) ?
Email (string) ?
Fax (string) ?
Flower (string) ?
Grapevine (string) ?
Hat (short) ?
Kind (string) ?
MichelinCat (string) ?
Name (string)
    Veronelli.Restaurant.Name (string)
    ACenaAccademia.Restaurant.Name (string)
    ACenaRistEuropaPaolini.Restaurant.Name (string)
    ACenaEspresso.Restaurant.Name (string)
    ACenaGamberoRosso.Restaurant.Name (string)
    Michelin.Restaurant.Name (string)
Parking (string) ?
```

Not Mapped Attributes:

```
Sources
```

Mapping Table

Restaurant	Restaurant(Veron...	Restaurant(ACena...	Restaurant(ACena...	Restaurant(ACena...	Restaurant(ACena...	Restaurant(Michel...
Address	Address	Address	Address	Address	Address	Address
Booking	Booking					Equipment
Category		Category				
Cheese	Cheese					

Figure 7. The Mapping Table for the Restaurant Global Class

3. THE MOMIS QUERY MANAGER

The MOMIS Query Manager is a coordinated set of functions which takes an incoming query (say global query), it defines a decomposition of the query according to the mapping of the GVV onto the local data sources relevant to the query, send the subqueries to these data sources, collects their answers, performs any residual filtering as necessary, and finally delivers the answer to the enquiring user.

The processing of queries expressed in the GVV (*global query*) consists of the following steps:

1) *Query rewriting*: to rewrite a global query as an equivalent set of queries expressed in the local sources (*local queries*).

2) *Local queries execution*: the local queries are sent and executed at local sources.

3) *Fusion and Reconciliation:* the local answers are fused into the global answer.

3.1 Query rewriting

The query rewriting method depends on the approach used to model the mappings between the GVV and the local schemata: MOMIS uses a global-as-view (GAV) approach (Halevy, 01), then the global query is rewritten by means of unfolding, that is, by expanding each atom of the global query according to its definition in the mapping. In the following example we describe our query rewriting method.

Given a global class G, related to the local class L1, L2, …, Ln, we consider a Global Query Q over G:

```
Q = select <Q_select-list> from G where <Q_condition>
```

where <Q_condition> is a Boolean expression of positive atomic constraints: (GA1 op value) or (GA1 op GA2), where GA1 and GA2 are attributes of the global class G. Note that we do not consider negation in our framework.

The *query rewriting* process is made up of the following steps:

1) Atomic constraint mapping

In this step, each atomic constraint of Q is rewritten into one that can be supported by the local class. The atomic constraint mapping is performed on the basis of *mapping functions* defined in the Mapping Table: for each element MT[GA][L] that is not a null we define a *mapping function*, denoted by MT_F[GA][L], which represents how the local attributes of L are mapped to the global attribute GA.

The function MT_F[GA][L] must be a function that is executable/supported by the local source of the class L. For example, for relational sources, MT_F[GA][L] is an SQL value expression; the following defaults hold: if MT[GA][L]={LA} then MT_F[GA][L]=LA and, if MT[GA][L] contains more than one string attribute, then MT_F[GA][L] is the string concatenation.

The constraint mapping depends on the definition of *Resolution Functions* for global attributes, which will be introduced in subsection 3.3; an example of constraint mapping will be shown in section 3.4.

2) Residual Constraints computation

Intuitively, residual constraints are the constraints of the global query that are not mapped onto all local queries.

3) Local select-list computation

The select-list of a local query is a set of attributes, including the global query attributes, the join attributes, the residual constraints attributes, and they are translated into the corresponding set of local attributes on the basis of the mapping table.

The output of the query rewriting process is a set of local queries; each local query Q_L over L is in a form that is supported on the local source of the class L. For example, for relational sources, a local query Q_L over L will look like this:

```
Q_L =      select <Q_L_select-list> from L where <Q_L_condition>
```

3.2 Local queries execution

A local query L_Q is sent to the source including the local class L; its answer is transformed by applying the mapping functions related to L: in this way, we perform the conversion of the local class instances into the GVV instances. The result of this conversion is materialized in a temporary table.

3.3 Fusion and Reconciliation

An important task of a mediator is to perform *object fusion*, that is grouping together information about the same real-object stored in different data sources. When all the local query answers have materialized, we fuse and reconcile them into the global answer.

Merging data from different sources requires different representations of the same real world object to be identified; this process is called *object identification* (Naumann, 02). The topic of object identification is currently a very active area of research and there is significant contribution both from the artificial intelligence (Tejada, 01) and database communities (Ananthakrishna, 02, Chaudhuri , 03) on this problem.

3.3.1 **Join Conditions**

To identify instances of the same object and fuse them we introduce *Join Conditions* among pairs of local classes belonging to the same global class. Given two local classes L1 and L2 belonging to G, a Join Condition between L1 and L2, JC(L1,L2), is an expression over L1.Ai and L2.Aj where Ai (Aj) are global attributes[1] with a not null mapping in L1 (L2).

In order to allow the designer to define a join condition which holds up for each pair of local classes belonging to a Global Class G, we introduce the *Join Map* for G, denoted by JM(G), which is an expression JM(G) = ϕ(X,Y) over X.Ai and Y.Aj, where X and Y denote local classes and Ai and Ai, called *Join Attribute*, are global attributes with a not null mapping in all the local classes. In this way, the Join Condition between each pair of L1 and L2 in G is obtained as ϕ(L1,L2). In the actual implementation of the system, JM(G) is an SQL expression.

The designer can explicitly define the Join Map JM(G) or can use some predefined expressions; for default, selecting JA1, …, JAn as Join Attributes the following *equality join condition* is defined as a Join Map:

EJC(X,Y) : X.JA1 = Y.JA1 and … and X.JAn = Y.JAn

For example, in the Restaurant Global Class we could select the attributes Name and City as join attributes. In this way, when two objects in our information system have the same Name and City, they represent the same restaurant. This fact is true independently from the way (face, color, size, case) in which this information is stored in the database (e.g. it is possible to identify the data about the ‘Fini’ restaurant in ‘Modena’ within the ‘Veronelli’, ‘Michelin’ and ‘Ristoranti d’europa’, regardless of the way used to write the name).

On the other hand, this Join Condition would not be sufficient to identify the same restaurant in different databases. For example, there are restaurants named in different way in different guides, but located in the same place: ‘La Francescana’ in Modena is called in this way in the ‘Veronelli’ guide, but in the Michelin guide there is a restaurant in Modena named ‘Osteria la Francescana’. These two database entries actually are the same restaurant.

In this case the designer may apply a new operator, called *containment join condition (CJC)*, defined by exploiting the LIKE operator in order to check the containment of the values of two string attributes. For example, by selecting Name, City as Join Attributes the operator is defined as follows:

CJC(X,Y) : (‘%’ + X.Name + ‘%’ LIKE ‘%’ + Y.Name + ‘%’ AND X.City = Y.City) OR

(X.Name = Y.Name AND ‘%’ + X.City + ‘%’ LIKE ‘%’ + Y.City + ‘%’)

By using CJC(X,Y) as a Join Map, the system recognizes that ‘La Francescana’ and ‘Osteria la Francescana’ are the same real world object.

When the predefined expressions are too vague, the designer can define ad hoc functions with more restrictive conditions. For example:

CJC(X,Y) : ‘%’ + X.Name + ‘%’ LIKE ‘%’ + Y.Name + ‘%’ AND X.City = Y.City

3.3.2 **Full-Disjunction**

In our GAV approach, each global class is expressed by means of the *Full Disjunction* (Galindo-Legaria, 94) that has been recognized as providing a natural semantics for data merging queries

[1] Note that a join condition is an expression over global attributes: in fact, this condition is applied after the conversion of the local class instances into the GVV instances.

(Rajaraman, 96). The informal definition of Full Disjunction is the following: "Computing the natural outerjoin of many relations in a way that preserves all possible connections among facts". We apply this definition to our context, then, given a global class G composed of L1, L2, …, Ln, the instance of G is the full-disjunction of L1, L2, .. , Ln, computed on the basis of the Join Conditions; intuitively, we use the Join Conditions instead of the natural outer join. In the case of two classes, the Full Disjunction corresponds to the full outer join.

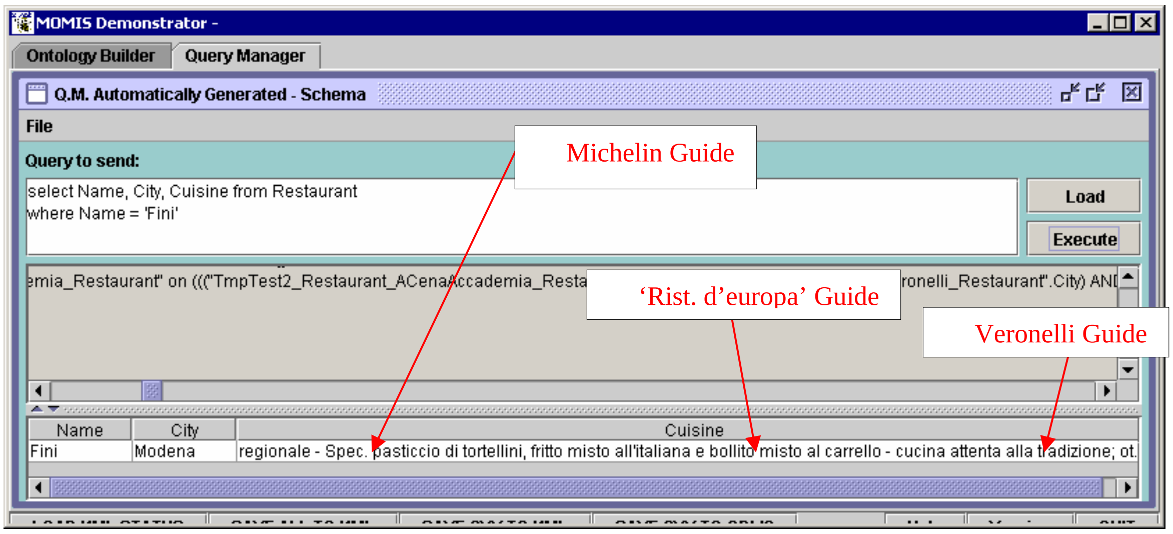
3.3.3 **Resolution functions**

a Resolution Function (Naumann02) may be defined For each global attribute mapping onto local attributes coming from several local source, in order to solve data conflicts due to different local attribute values; in this way we can define what value shall appear in the result. Our system provides some standard kinds of resolution functions:

- Random function: this function results in having the content of the one of the local attribute randomly chosen.
- Aggregation functions for numerical attributes: SUM, AVG, MIN, MAX, …
- Precedence function: The highest informational quality value on the basis of an information quality model.
- Coalesce function: The first not null value among the local attributes values is the function result.

In order to develop and implement the tourist information system we developed and customized some specific functions:

1) **All-values**. Within the web sites, in general there is an area devoted to the description of the "Cuisine" kind. Different web sites describe the same cuisine in different ways. Considering the integration viewpoint, the cuisine description provided by each site is globally considered by means of a unique global attribute (Cuisine). We may apply one of the previously defined functions to this global attribute (e.g. the random function in order to give us a result one of the local descriptions randomly chosen for each query), or we may define a new function allowing the user to have all the data provided by the local sources. This new function is called *all-values*, and its application in our example for the 'Fini' restaurant in Modena gives as a result different values of that attribute from the 'Veronelli', 'Michelin' and 'Ristoranti d'europa' guides which gives us the following result:



```
select Name, City, Cuisine from Restaurant
where Name = 'Fini'
```

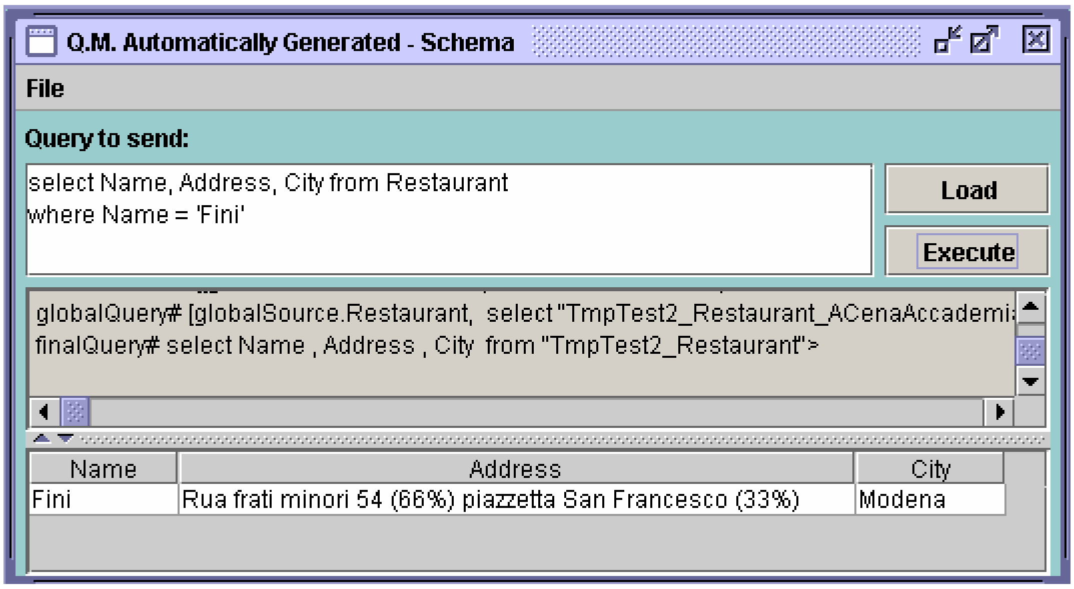
```
emia_Restaurant" on ((("TmpTest2_Restaurant_ACenaAccademia_Resta ... ronelli_Restaurant".City) ANI
```

Name	City	Cuisine
Fini	Modena	regionale - Spec. pasticcio di tortellini, fritto misto all'italiana e bollito misto al carrello - cucina attenta alla tradizione; ot.

Figure 8. An example of the all values resolution function

2) **Rate-values**. Sometimes many local sources are integrated and then it is not useful or readable to have a unique global attribute value, especially if more sources provide the same values. The *rate-values* function is a specialization of the *all-values* function and it groups equal values into one providing it with a score indicating the amount of sources where this specific value is found. Building the tourist information system, we discovered that sometimes the local sources provided different addresses for the same restaurant (e.g. the restaurant changed the place or the town hall changed the

name/number of that street, and not all local sources where updated). In this case, only a few sources provide a result different from the other ones. The following figure shows the function applied to the address global attribute for the 'Fini' restaurant.



```
select Name, Address, City from Restaurant
where Name = 'Fini'
```

```
globalQuery# [globalSource.Restaurant,  select "TmpTest2_Restaurant_ACenaAccademi
finalQuery# select Name , Address , City  from "TmpTest2_Restaurant">
```

Name	Address	City
Fini	Rua frati minori 54 (66%) piazzetta San Francesco (33%)	Modena

Figure 9. An example of the rate values resolution function

3.4 Homogeneous Attributes

If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*. Of course, we do not need to define a Resolution Function for homogeneous attributes. A global attribute mapped onto one source is a particular case of homogeneous attribute.

3.4.1. Atomic constraint mapping for Homogeneous Attribute

The constraint mapping depends on the definition of the Resolution Function, for example, if the numerical global attribute GA is mapped onto L1 and L2, and we define AVG function as resolution function, the constraint (GA = value) cannot be pushed at the local sources, because of the AVG function has to be calculated at a global level, the constraint may be globally true but locally false. In this case, the constraint is mapped as *true* in both the local sources. On the other hand, if GA is an homogeneous attribute the constraint can be pushed at the local sources.

For homogeneous attributes the constraint mappings are defined as follows:

- An atomic constraint (GA op value) is mapped onto the local class L as:

(MT_F[GA][L] op value)	**if**	MT[GA][L] is not null and the op operator is supported into L
true	**otherwise**	

- An atomic constraint (GA1 op GA2) is mapped into the local class L as:

(MT_F[GA1][L] op MT_F[GA2][L])	**if**	MT[GA1][L] and MT[GA1][L] are not null and the op operator is is supported into L
true	**otherwise**	

By default, each operator used in the global query is supported into a local class: the designer can define, the operators which are not supported for each local class. The current implementation of the system assumes that each operator, OP, used in the global query is supported into a local class, i.e. a constraint including OP can be solved in local class. As a future work, we plan to extend our framework by considering a general mechanism for translating constraint queries such as the one proposed in (Chang, 99).

For non homogeneous attributes the constraint mappings must be performed on the basis of the resolution functions; in general, an atomic constraint defined for non homogeneous attributes cannot be rewritten in the local sources and is rewritten as true (this is the method currently adopted in our system).

4. MOMIS as a Web service mediator data provider

The MOMIS GVV and query manager is encapsulated into three different web services in order to improve the interoperability with existent applications.

In a typical Web services architecture (Ferris, 03), a service provider has a service that is made available to other systems to use independently no matter what kind of platforms, operating systems and programming languages are used. This is achieved by using WSDL, the Web Service Description Language, a proposed standard that provides a model and an XML format to describe web services. WSDL allows the separation of the description of the abstract functionality offered by a service, from concrete details of a service description (e.g. "how" and "where" the functionality is offered) (W3C, 01). The provider creates a WSDL service description that defines the service interfaces, the functionalities of the service and the input and output messages for each operation. A typical exchange scenario coming from a Web service call produces the following steps: (i) execute the service to the provider and produce relevant XML documents from source data and, (ii) ship the produced documents to the requester that has requested them.

In Figure 10 we show the three different Web Services implemented by means of SOAP and WSDL as description language: Information Service, Metadata Service and Search Service.

The Information Service provides for each GVV, its name and the types and descriptions of the integrated data sources, i.e. restaurant guides or catalogues that are managed by the system.

The Metadata Service provides all the obtained semantic mappings existing among restaurants schemas; moreover, it provides the annotation of the involved schemas, i.e., the meaning of classes with respect to the common lexical ontology Wordnet.

The Data Search Service provides the query engine functionalities on the integrated schemas; it delivers the query answers by means of an XML file following a template DTD. This service establishes a very easy integration with a web application, where the business level calls the web service that returns to the business logic the query acknowledgement and the URI of the resulting XML file: then the business logic applies the desired XSL stylesheet and dynamically produces a web page reporting the information.

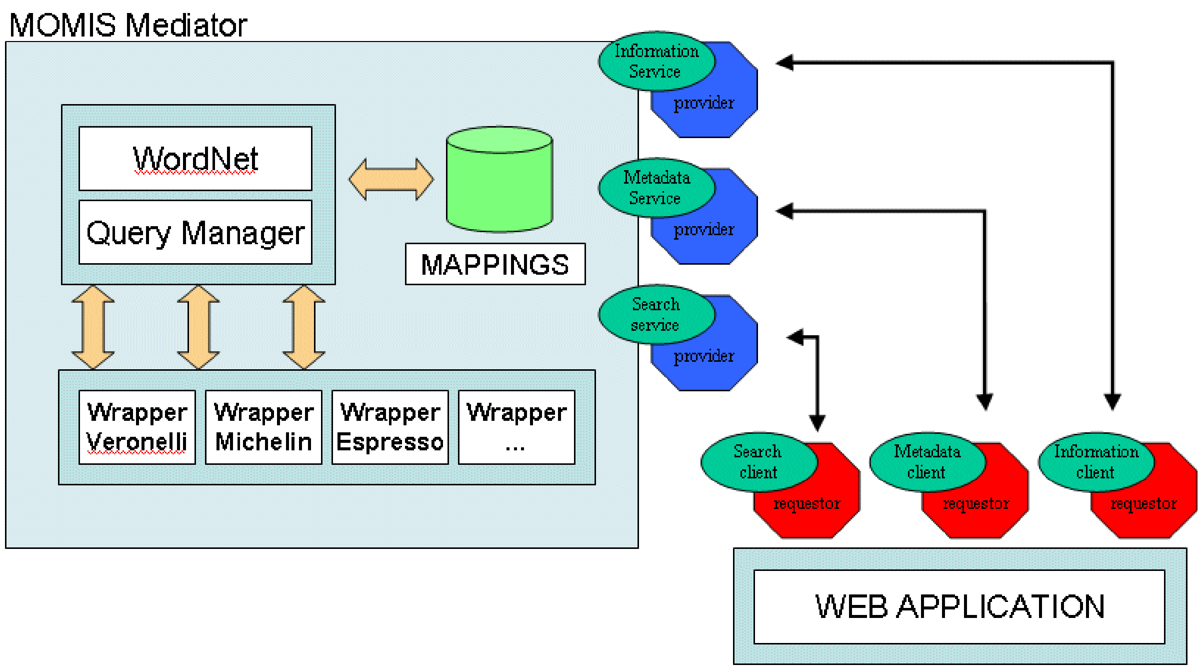


Figure 10. The MOMIS web services architecture

5. RELATED WORK

In the area of heterogeneous information integration, many projects based on mediator architectures have been developed. These works describe general approaches and the application of the proposed architectures to a specific domain (Baumgartner, 01, Nodine, 03). In this paper we described the application of the MOMIS system for both integrating data sources concerning the tourist domain and giving the possibility of querying them. The result of this work is the creation of a complete tourist information system which can interoperate with existing Internet applications.

Concerning the integration area, the mediator-based TSIMMIS project (Li, 98) follows a "structural" approach and uses a self-describing model (OEM) to represent heterogeneous data sources and the MSL (Mediator Specification Language) rule to enforce source integration. In TSIMMIS, by means of MSL, arbitrary views (in particular, recursive views) can be defined at the mediator layer. The MOMIS system made a different choice: starting from the semi-automatic generated mappings between global and local attributes stored in the mapping tables, views (global classes) are defined by means of a predefined operator, i.e. the full disjunction, that has been recognized as providing a natural semantics for data merging queries. In particular, in the view definition resolution functions are defined to take into account data conflicts.

The SIMS (Knoblock, 00) proposes the creation of a global schema definition by exploiting the use of Description Logics (i.e., the LOOM language) for the description of information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.

The Information Manifold system (Levy, 98) provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources and the integrated schema is mainly defined manually by the designer, while in our approach it is tool-supported.

The goal of Clio (Yan, 01), (Miller, 01) is to develop a tool for semi-automatically creating mappings between two data representations (i.e., with user input). First of all, in the Clio framework the focus is on the schema mapping problem in which a source is mapped onto a different, but fixed, "target" schema, while the focus of our proposal is the semi-automatic generation of a "target" schema, i.e. the Global Virtual View, starting from the sources. Moreover, the semi-automatic tool for creating schema mapping, developed in Clio, employs a mapping-by-example paradigm that relies on the use of value mappings describing how a value of a target attribute can be created from a set of values of source attributes. Our proposal for creating schema mappings can be considered orthogonal with respect to this paradigm. In fact, the main techniques of mapping construction rely on the meanings of the class and attribute names selected by the designer in the annotation phase and by considering the semantic relationships between meanings coming from the common lexical ontology. On the other hand, MOMIS and CLIO share a common mapping semantics among a (target) global schema and a set of source schemata expressed by the full-disjunction operator.

Infomaster (Genesereth, 97) provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. The main difference of this project w.r.t. our approach is the lack of a tool aid-support for the designer in the integration process.

Also some work has been done on integration systems inside the multi-agent system community. a particular mention is due to the InfoSleuth system (Nodine, 00)which has some similarities with MOMIS system goal, . InfoSleuth is a system designed to actively gather information by performing diverse information management activities. InfoSleuth agents enable a loose integration of technologies allowing: (1) extraction of semantic concepts from autonomous information sources; (2) registration and integration of semantically annotated information from diverse sources; and (3) temporal monitoring, information routing, and identification of trends appearing across sources in the information network.

Another important experience is the RETSINA multi-agent infrastructure for in-context information retrieval (Sycara 99). In particular the LARKS description language is defined to realize the agent matchmaking process (both at syntactic and semantic level) by using several different filters: Context, Profile, Similarity, Signature and Constraint matching.

Inside the e-tourism research area intelligent restaurant recommender systems have been developed (Tung, 04), (Goren-Bar, 04). These systems try to suggest the best place for the client on the basis of some user preferences and a location. Nevertheless, they are mainly focused on the user location by using GPS and the wide range of mobile devices where the systems run rather than the information integration process. Among the tourist application, the most similar to our approach is TheaterLoc (Barish, 00), an information integration application that allows users to retrieve information about theaters and restaurants for a variety of cities in the United States. Like our approach, TheaterLoc follows a wrapper/mediator technology and integrates five different heterogeneous and distributed sources, one for each different topic (Restaurant, Theater, Trailer, Geocoder, …), hence the information overlapping is not very relevant as in our Restaurant application.

6. CONCLUSIONS

We described a semi-automated approach to information extraction and integration ,that has been implemented in the MOMIS system following conventional wrapper/mediator architecture. The Ontology Builder tool interfaces all the employed modules with the goal of allowing an interactive and customized use of MOMIS techniques by the designer, based on the specific requirements of a given integration process. The Query Manager implements techniques addressed to achieve two main goals: *optimal query reformulation* w.r.t. local sources and *object fusion*, i.e. grouping together information (from the same or different sources) about the same real-world entity.

Furthermore, we presented the new Web Services-based architecture of the MOMIS framework. The architecture enhances the semantic integration features of the MOMIS leveraging new technologies such as XML Web Services and the SOAP protocol.

References

(Ananthakrishna, 02) R. Ananthakrishna, S. Chaudhuri, and V. Ganti: Eliminating fuzzy duplicates in data warehouses. VLDB 2002: 586-597

(Barish, 00) G. Barish, C. A. Knoblock, Y. S. Chen, S. Minton, A. Philpot, C. Shahabi: The TheaterLoc Virtual Application. AAAI/IAAI 2000: 980-987.

(Baumgartner, 01) R. Baumgartner, S. Flesca, Georg Gottlob: Supervised Wrapper Generation with Lixto. VLDB 2001: 715-716

(Benassi, 04) R. Benassi, S. Bergamaschi, A. Fergnani, D. Miselli: "Extending a Lexicon Ontology for Intelligent Information Integration", European Conference on Artificial Intelligence (ECAI2004). Valencia, Spain, 22-27 August 2004.

(Beneventano, 97) D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini “ODB-QOptimizer: a tool for semantic query optimization in OODB”. ICDE’97, UK, April 1997.

(Beneventano, 98) D. Beneventano, S. Bergamaschi, S. Lodi, C. Sartori "Consistency Checking in Complex Object Database Schemata with Integrity Constraints" IEEE Transactions on Knowledge and Data Engineering, 10(4):576-598, 1998

(Beneventano, 03) D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini: “Synthesizing an Integrated Ontology”. IEEE Internet Computing 7(5): 42-51 (2003).

(Beneventano, 03b) D. Beneventano, S. Bergamaschi, C. Sartori: "Description Logics for Semantic Query Optimization in Object-Oriented Database Systems", ACM Transaction on Database Systems, Volume 28: 1-50 (2003).

(Bergamaschi, 01) S. Bergamaschi, S. Castano, D. Beneventano, M. Vincini: "Semantic Integration of Heterogeneous Information Sources", Special Issue on Intelligent Information Integration, Data & Knowledge Engineering, Vol. 36, Num. 1, Pages 215-249, Elsevier Science B.V. 2001.

(Castano, 01) S. Castano, V. De Antonellis, S. De Capitani di Vimercati: Global Viewing of Heterogeneous Data Sources. IEEE Trans. Knowl. Data Eng. 13(2): 277-297 (2001)

(Cattell, 00) R. G. G. Cattell, Douglas K. Barry: “The Object Data Standard: ODMG 3.0”, Morgan Kaufmann, 2000.

(Chang, 99) Kevin Chen-Chuan Chang, Hector Garcia-Molina: Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources. SIGMOD Conference 1999: 335-346.

(Chaudhuri , 03) S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani: Robust and efficient fuzzy match for online data cleaning. SIGMOD '03, 2003.

(Ferris, 03) C. Ferris, J. Farrell. What Are Web Services? Communications of the ACM 46(6), June 2003.

(Galindo-Legaria, 94) C. A. Galindo-Legaria, “Outerjoins as Disjunctions”. SIGMOD Conference 1994, 348-358.

(Genesereth, 97) M. R. Genesereth, A. M. Keller, O. Duschka, “Infomaster: An Information Integration System”, in proceedings of 1997 ACM SIGMOD Conference, May 1997.

(Goren-Bar, 04) D. Goren-Bar, T. Kuflik: “Don't miss-r -: recommending restaurants through an adaptive mobile system”. In proc. of Intelligent User Interfaces 2004: 250-252.

(Gottlob, 04) G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, S. Flesca: “The Lixto Data Extraction Project - Back and Forth between Theory and Practice”. PODS 2004: 1-12.

(Halevy, 01) A. Halevy, A. Y. Halevy. “Answering queries using views: A survey”. Very Large Database J., 10(4):270–294, 2001.

(Knoblock, 00) Knoblock C. A. Ambite J.L. “Flexible and scalable cost-based query planning in mediators: A transformational approach”. Artificial Intelligence, 118(1-2):115-161,
2000.

(Laender, 02) A. H. F. Laender, B. Ribeiro-Neto, and A. S. da Silva. “DEByE: Data Extraction By Example". Data and Knowledge Engineering, 40(2):121-154, Feb. 2002.

(Levy, 98) A. Levy, "The Information Manifold Approach to Data Integration," IEEE Intelligent Systems, 1312-16, 1998.

(Li, 98) C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman, M. Valiveti. “Capability Based Mediation in TSIMMIS”. SIGMOD 98, Seattle, June 1998.

(Liu, 00) L. Liu, C. Pu, and W. Han. “XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources". In Proc. ICDE 2000, pages 611-621, San Diego, USA, 2000.

(Ludascher, 98) B. Ludäscher, R. Himmeröder, G. Lausen, W. May and C. Schlepphorst. “Managing Semistructured Data with Florid: A Deductive Object-oriented Perspective". Information Systems, 23(8):1-25, 1998.

(Miller, 95) A.G. Miller. “A lexical database for English”. Communications of the ACM, 38(11):39:41,1995.

(Miller, 01) R. J. Miller, M. A. Hernandez, L. M. Haas, L. Yan, C. T. H. Ho, L. Popa, and R. Fagin, “The Clio project: managing heterogeneity”, ACM SIGMOD Record 30, 1 (March 2001), pp. 78-83.

(Naumann, 02) F. Naumann, M. Häussler: “Declarative Data Merging with Conflict Resolution”. International Conference on Information Quality (IQ 2002). 2002, pages 212-224.

(Nodine, 00) M. H. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. C. Taylor, A.Unruh: “Active Information Gathering in Infosleuth”. IJCIS 9(1-2): 3-28 (2000).

(Nodine, 03) M. H. Nodine, A. H. H. Ngu, A. R. Cassandra, W. Bohrer: Scalable Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth^{TM}. IEEE Trans. Knowl. Data Eng. 15(5): 1082-1098 (2003)

(Rajaraman, 96) A. Rajaraman , J. D. Ullman: “Integrating Information by Outerjoins and Full Disjunctions”. PODS 1996, pages 238-248

(Tejada, 01) S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. Information Systems Journal Special Issue on Data Extraction, Cleaning, and Reconciliation, 2001, 26(8): 607-633

(Tung, 04) H. W. Tung, V. W. Soo: “A Personalized Restaurant Recommender Agent for Mobile E-Service”, IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE 04), 29-31 March 2004, Taipei, Taiwan. IEEE Computer Society 2004, ISBN 0-7695-2073-1

(Sahuguet, 01) A. Sahuguet and F. Azavant. “Building Intelligent Web Applications Using Lightweight Wrappers". Data and Knowledge Engineering, 36(3):283-316, 2001.

(Sycara, 99) K. Sycara, “In-context information management trough adaptative collaboration of intelligent agents”, Intelligent Information Agents, 78-99, 1999.

(W3C, 01) W3C, Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001.

(Yan, 01) L. Yan, R. J. Miller, L. M. Haas, and R. Fagin, “Data-driven understanding and refinement of schema mappings”, Proc. 2001 ACM SIGMOD Conference (SIGMOD '01), pp. 485-496.