Mobile Agents for Information Integration

S. Bergamaschi[1][2], G. Cabri[1], F. Guerra[1], L. Leonardi[1], M. Vincini[1], and F. Zambonelli[1]
e-mail: {sonia.bergamaschi,giacomo.cabri,francesco.guerra,letizia.leonardi, maurizio.vincini,franco.zambonelli}@unimo.it

[1] Università di Modena e Reggio Emilia
DSI - Via Vignolese 905, 41100 Modena
[2] CSITE-CNR Bologna
V.le Risorgimento 2, 40136 Bologna

Abstract. The large amount of information that is spread over the Internet is an important resource for all people but also introduces some issues that must be faced. The dynamism and the uncertainty of the Internet, along with the heterogeneity of the sources of information are the two main challanges for the today's technologies. This paper proposes an approach based on mobile agents integrated in an information integration infrastructure. Mobile agents can significantly improve the design and the development of Internet applications thanks to their characteristics of autonomy and adaptability to open and distributed environments, such as the Internet. MOMIS (Mediator envirOnment for Multiple Information Sources) is an infrastructure for semi-automatic information integration that deals with the integration and query of multiple, heterogeneous information sources (relational, object, XML and semi-structured sources). The aim of this paper is to show the advantage of the introduction in the MOMIS infrastructure of intelligent and mobile software agents for the autonomous management and coordination of the integration and query processes over heterogeneous data sources.

1 Introduction

Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information systems for cooperation and interoperability. In the past, companies have equipped themselves with data storing systems building up informative systems containing data that are related one another, but which are often redundant, heterogeneous and not always substantial. The problems that have to be faced in this field are mainly due to both structural and application heterogeneity, as well as to the lack of a common ontology, causing semantic differences between information sources. Moreover, these semantic differences can cause different kinds of conflicts, ranging from simple contradictions in name use (when different names are used by different sources to indicate the same or similar real-world concept), to structural conflicts (when different models/primitives are used to represent the same information). Complicating factors with respect to conventional view integration techniques [3] are related to the fact that semantic heterogeneity occurs on the large-scale. This heterogeneity involves terminology, structure, and domain of the sources, with respect to geographical, organizational, and functional aspects of the information use [28]. Moreover, to meet the requirements of global, Internet-based information systems, it is important that the tools developed for supporting these activities are semi-automatic and scalable as much as possible.

To face the issues related to scalability in the large-scale, in this paper we propose the exploitation of *mobile agents* in the information integration area, and, in particular, their integration in the MOMIS infrastructure, which focuses on capturing and reasoning about semantic aspects of schema descriptions of heterogeneous information sources for supporting integration and query optimization.

Mobile agents are a quite recent technology. They can significantly improve the design and the development of Internet applications thanks to their characteristics. The agency feature [25] permits them to exhibit a high degree of autonomy with regard to the users: they try to carry out their tasks in a *proactive* way, *reacting* to the changes of the environment they are hosted. The mobility feature [26] takes several advantages in a wide and unreliable environment such as the Internet. First, mobile agents can significantly save bandwidth, by moving locally to the resources they need and by carrying the code to manage them. Moreover, mobile agents can deal with non-continuous network connection and, as a consequence, they intrinsically suit mobile computing systems. All these features are particularly suitable in the information retrieval area [10].

MOMIS [8, 5, 7] (Mediator envirOnment for Multiple Information Sources) is an infrastructure for information integration that deals with the integration and query of multiple, heterogeneous information sources, containing structured and semistructured data. MOMIS is a support system for semiautomatic integration of heterogeneous sources schema (relational, object, XML and semistructured sources); it carries out integration following a semantic approach which uses Description logics-based techniques, clustering techniques and an ODM-ODMG [15] extended model to represent extracted and integrated information, ODM$_{I^3}$. Using the ODL$_{I^3}$ language, referred to the ODM$_{I^3}$ model, it is possible to describe the sources (local schema) and MOMIS supports the designer in the generation of an integrated view of all the sources (Global Virtual View), which is expressed using XML standard. The use of XML in the definition of the Global Virtual View lets to use MOMIS infrastructure with other open integration information systems by the interchange of XML data files.

In particular, we show the advantage of the introduction in the MOMIS architecture of intelligent and mobile software agents for the autonomous management and coordination of the integration and query processes over heterogeneous data sources.

The outline of the paper is the following. Section 2 presents the MOMIS system architecture and the role the mobile agents. Section 3 contains the MOMIS approach to data integration. Section 4 presents some related work. Finally Section 5 reports the conclusions and sketches future work.

2 System Architecture References

Like other integration projects [1, 32], MOMIS follows a “semantic approach” to information integration based on the conceptual schema, or metadata, of the information sources, and on the the I^3 architecture [24] (see Figure 1). The system is composed by the following components:

1. a common data model, ODM$_{I^3}$, which is defined according to the ODL$_{I^3}$ language, to describe source schemas for integration purposes. ODM$_{I^3}$ and ODL$_{I^3}$ have been defined in MOMIS as subset of the corresponding ones in ODMG, following the proposal for a standard mediator language developed by the I^3/POB working group [9]. In addition, ODL$_{I^3}$ introduces new constructors to support the semantic integration process;
2. *Wrapper agents*, placed over each sources, translate metadata descriptions of the sources into the common ODL$_{I^3}$ representation, translate (reformulate) a global query expressed in the OQL$_{I^3}$[1] query language into queries expressed in the sources languages and export query result data set;
3. a *Mediator*, which is composed of two components in its turn: the *SI-Designer* and the *Query Manager* (QM). The SI-Designer component processes and integrates ODL$_{I^3}$ descriptions received from wrapper agents to derive the integrated representation of the information sources. The QM component performs query processing and optimization. Starting from each query posed by the user on the Global Schema, the QM generates OQL$_{I^3}$ queries that are sent to wrapper agents by exploiting *query agents*, which are mobile agents. QM automatically generates the translation of the query into a corresponding set of sub-queries for the sources and synthesizes a unified global answer for the user.

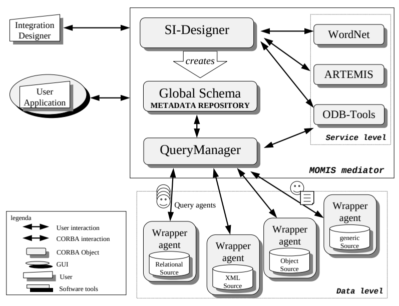

Fig. 1. The MOMIS system architecture

The original contribution of MOMIS is related to the availability of a set of techniques for the designer to face common problems that arise when integrating pre-existing information sources, containing both semistructured and structured data.
MOMIS provides the capability of explicitly introducing many kinds of knowledge for integration, such as integrity constraints, intra- and inter-source intensional and extensional relationships, and designer supplied domain knowledge. A *Common Thesaurus*, which has the role of a shared ontology of the source is built in a semi-automatic way. The *Common Thesaurus* is a set of intra and inter-schema intensional and extensional

[1] OQL$_{I^3}$ is a subset of OQL-ODMG.

relationships, describing the knowledge about classes and attributes of sources schemas; it provides a reference on which to base the identification of classes candidate to integration and subsequent derivation of their global representation.

MOMIS supports information integration in the creation of an integrated view of all sources (Global Virtual View) in a way automated as much as possible and performs revision and validation of the various kinds of knowledge used for the integration. To this end, MOMIS combines reasoning capabilities of Description Logics with affinity-based clustering techniques, by exploiting a common ontology for the sources constructed using lexical knowledge from WordNet [23, 29] and validated integration knowledge.

The Global Virtual View is expressed by using XML standard, to guarantee the interoperability with other open integration system prototype.

2.1 The Roles of the Agent

In our architecture, agents have two main roles. On the one hand, the source wrappers are agents that converts the source information and react to source changes. On the other hand, the QM exploits mobile agents to carry out the retrieval of information.

When a new source has to be integrated in MOMIS, a mobile agent moves to the site where the source resides. Such agent checks the source type and autonomously installs the needed driver to convert the source information. Moreover, a fixed agent, called *wrapper agent*, is left at this site to preside the source. Besides interacting with query agents as described later, the wrapper agents monitor the changes that may occur in the data structure of sources; when a change occurs in a source, the corresponding wrapper agent creates an appropriate mobile agent that moves to the MOMIS site to inform about the new structure, so as to update the Global Schema.

The QM works as follows. When a user queries the global schema, it generates a set of sub-queries to be made to the single sources. To this purpose, it can exploit one or more *query agents*, which move to the source sites where they interact with the wrapper agents. The choice of the number of query agents to use can be determined by analyzing each query. In some cases, it is better to delegate the search to a single query agent, which performs a “trip” visiting each source site: it can start from the source that is supposed to reduce the further searches in the most significant way, then continue to visit source sites, performing queries on the basis of the already-found information. In other cases, sub-queries are likely to be quite independent, so it is better to delegate several query agents, one for each source site: in this way the searches are performed concurrently with a high degree of parallelism. In any case, the peculiar features of mobile agents are exploited and make the MOMIS searches suitable to the Internet environment. First, by moving locally to the source site, a query agent permits to significantly save bandwidth, because it is not necessary to transfer a large amount of data, but the search computation is performed locally where the data resides. Second, MOMIS can queries also sources that do not have continuous connections: the query agent moves to the source site when the connection is available, performs locally the search even if the connection is unstable or unavailable, and then returns to the QM site as soon as the connection is available again. Finally, this fits well mobile computing, where mobile devices (which can host users, MOMIS, or sources) do not have permanent connections with the fixed network infrastructure.

```
<!ELEMENT fiat(car*)>
<!ELEMENT car(name,engine,dimensions,tires,
      performance,price)>
<!ELEMENT engine(name,cylinders?,layout?,
      capacity_cc?,compression_ratio?,
      power_kw, fuel_system)>
<!ELEMENT dimensions(length,width,heigth,
      luggage_capacity)>
<!ELEMENT performance (urban_consumption,
      combined_consumption,speed)>
<!ELEMENT name (#pcdata)>
...
```

Fig. 2. Fiat database (FIAT)

The interaction between agents can occur by using several protocols. See [12] for a comparison among different kinds of coordination for Internet applications based on mobile agents; an approach that is gaining ground more and more is the one based on *programmable tuple spaces* [11].

2.2 Running Example

```
Vehicle(name, length, width, height)
Motor(cod_m, type, compression_ratio,
      KW, lubrification, emission)
Fuel_Consumption(name, cod_m, drive_trains,
      city_km_l, highway_km_l )
Model(name, cod_m, tires, steering, price)
```

Fig. 3. Volkswagen database (VW)

In order to illustrate how the MOMIS approach works, we will use the following example of integration in the Car manufacturing catalogs, involving two different data-sources that collect information about vehicle. The first data-source is the `FIAT` catalog, containing semistructured XML informations about cars of the italian car factory (see Figure 2).

The second data-source is the `Volkswagen` database (`VW`) reported in Figure 3, a relational database containing information about this kind of car. Both database schemata are built by analyzing the web site of this factory.

3 Integration Process

The MOMIS approach to perform Global Virtual View is articulated in the following phases:

1. *Generation of a Common Thesaurus*.
   The *Common Thesaurus* is a set of terminological intensional and extensional relationships, describing intra and inter-schema knowledge about classes and attributes of sources schemas. We express intra and inter-schema knowledge in form of terminological and extensional relationships (em synonymy, *hypernymy* and *relationship*) between classes and/or attribute names. In this phase, to extract lexicon derived relationships the WordNet database is used.
2. *Affinity analysis of classes*.
   Relationships in the *Common Thesaurus* are used to evaluate the level of *affinity* between classes intra and inter sources. The concept of affinity is introduced to formalize the kind of relationships that can occur between classes from the integration point of view. The affinity of two classes is established by means of affinity coefficients based on class names, class structures and relationships in *Common Thesaurus*.
3. *Clustering classes*.
   Classes with affinity in different sources are grouped together in clusters using hierarchical clustering techniques. The goal is to identify the classes that have to be integrated since describing the same or semantically related information.
4. *Generation of the mediated schema*.
   Unification of affinity clusters leads to the construction of the predicted schema. A class is defined for each cluster, which is representative of all cluster's classes and is characterized by the union of their attributes. The global schema for the analyzed sources is composed of all classes derived from clusters, and is the basis for posing queries against the sources.

In the following we introduce the generation of the *Common Thesaurus* associated with the example domain, starting from the lexicon relationships definition by using Wordnet.

3.1 Generation of a *Common Thesaurus*

The *Common Thesaurus* is a set of terminological intensional and extensional relationships, describing intra and inter-schema knowledge about classes and attributes of sources schemas; it provides a reference to define the identification of classes candidate to integration and subsequent derivation of their global representation. In the Common Thesaurus, we express knowledge in form of intensional relationships (SYN, BT, NT, and RT) and extensional relationships (SYN$_{ext}$, BT$_{ext}$, and NT$_{ext}$ between classes and/or attribute names.

The Common Thesaurus is constructed through an incremental process during which relationships are added in the following order:

1. *schema-derived relationships*: Intensional and extensional relationships holding at intra-schema level. These relationships are extracted analyzing each ODL$_{I^3}$ schema separately. In particular, intra-schema RT relationships are extracted from the specification of foreign keys in relational source schemas or a complex attributes (relationships) in object oriented database. When a foreign key is also a primary key

both in the original and in the referenced relation, a BT/NT relationship is extracted We show the most significant intra-schema relationship automatically generated from MOMIS:
⟨`VW.Model` RT `VW.vehicle`⟩
⟨`VW.Model` RT `VW.motor`⟩
⟨`fiat.engine` RT `fiat.car`⟩

2. *lexical-derived relationships*: Terminological relationships holding at inter- schema level are extracted by the SLIM module by analyzing different sources ODL$_{I^3}$ schemas together according to the Wordnet supplied ontology. Consider the `fiat` and the`VW` sources. The most significant lexical relationships derived using WordNet are the following:
   ⟨`fiat.car` SYN `VW.vehicle`⟩
   ⟨`fiat.engine.compression_ratio` SYN
   `VW.motor.compression_ratio`⟩
   ⟨`fiat.dimension` BT `VW.vehicle.width`⟩
3. *designer-supplied relationships*: Intensional and extensional relationships supplied directly by the designer, to capture specific domain knowledge about the source schemata. Consider the `VW` source, in which the model entity can be considered as a specialization of the vehicle entity. This relationship can not be automatically extracted using both the lexical and the structural approaches, hence we supplied the following relationship:
   ⟨`VW.Model` NT `fiat.car`⟩

   This is a crucial operation, because the new relationships are forced to belong to the Common Thesaurus and thus used to generate the global integrated schema. This means that, if a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema. Our system help the designer in detecting wrong relationships by performing a *Relationships validation* step with ODB-Tools. Validation is based on the compatibility of domains associated with attributes. For a complete description see [7].

   Referring to the *Common Thesaurus* resulting from our example, we show some significant relationships (for each relationships, control flag[1] denotes a valid relationship, while [0] an invalid one):
   ⟨`fiat.performance.combined_consumption` RT
   `vw.fuel_consumption.highway_km_l`⟩ [0]
   ⟨`fiat.dimensions` BT `vw.vehicle.height`⟩ [1]
   ⟨`VW.Model.name` RT `vw.vehicle.name`⟩ [1]

4. *inferred relationships*: Intensional and extensional new relationships, holding at intra-schema level, inferred by exploiting inference capabilities of ODB-Tools, a description logics based tool developed at University of Modena and Reggio Emilia [6]. In the examined domain ODB-Tools system infers the following relationships:
   ⟨`VW.Model` RT `fiat.dimensions`⟩
   ⟨`VW.Model` NT `fiat.engine`⟩

$\langle$`VW.motor` NT `fiat.car`$\rangle$

All these relationships are added to the Common Thesaurus and thus considered in the subsequent phase of construction of Global Schema. For a more detailed description of the above described process see [7].

Terminological relationships defined in each step hold at the intensional level by definition. Furthermore, in each of the above step the designer may "strengthen" a terminological relationships SYN, BT and NT between two classes C_1 and C_2 by establishing that they hold also at the extensional level, thus defining also an extensional relationship. The specification of an extensional relationship, on one hand, implies the insertion of a corresponding intensional relationship in the Common Thesaurus and, on the other hand, enable subsumption computation (i.e., inferred relationships) and consistency checking between two classes the C_1 and C_2.

Global Class and Mapping Tables

Starting from the output of the cluster generation, we define, for each cluster, a *Global Class* that represents the mediated view of all the classes of the cluster. For each global class a set of *global attributes* and, for each of them, the intensional mappings with the *local attributes* (i.e. the attributes of the local classes belonging to the cluster) are given [2].

Shortly, we can say that the global attributes are obtained in two steps: (1) Union of the attributes of all the classes belonging to the cluster; (2) Fusion of the "similar" attributes; in this step redundancies are eliminated in a semi–automatic way taking into account the relationships stored in the *Common Thesaurus*. For each global class a persistent *mapping-table* storing all the intensional mappings is generated; it is a table whose columns represent the set of the local classes which belong to the cluster and whose rows represent the global attributes.

The final step of the integration process provides the export of the Global Virtual View into a XML DTD, by adding the appropriate XML TAGs to represent the mapping table relationships. The use of XML in the definition of the Global Virtual View lets to use MOMIS infrastructure with other open integration information system by the interchange of XML data files. In addition, the Common Thesaurus is translated into XML file, so that MOMIS may provides a shared ontology that can be used by different semantic ontology languages [19, 18].

In the referring example the following *Global Class* are defined:

– Vehicle: contains the `Vehicle`, `Model`, `car` source classes and a global attributes indicates the source name;
– Engine: contains the `engine`, `Motor` source classes;
– Performance: contains the `performance`, `Fuel_Consumption` source classes;
– Dimensions: contains the `dimensions` source class.

[2] For a detailed description of the mappings selection and of the tool SI-Designer which assist the designer in this integration phase see [4].

Dynamic local schema modification

Now, let suppose that a modification occurs in a local source, for example in the FIAT DTD a new element `truck` is added:

```
<!ELEMENT truck(name, engine, dimensions, price, capacity)>
```

In this case the *local wrapper agent* that resides at the FIAT source creates a mobile agent that goes to the MOMIS site and there checks the permission to modify the Global Schema: if the agent is enabled it directly performs the integration phases (i.e. *Common Thesaurus*, *Clustering*, *Mapping Generation*) caused by the schema modification and notifies to the *Integration Designer* its actions. Otherwise, if the agent has not enough rights, it delegates the whole integration re-process to the *Integration Designer*.

In our example the new `truck` element is inserted by the em local wrapper agent in the Vehicle *Global Class* and the *mapping* is performed.

3.2 The Query Manager Agents

The user application interacts with MOMIS to query the Global Virtual View by using the OQL$_{I^3}$ language. This phase is performed by the QM component that generates the OQL$_{I^3}$ queries for wrapper agents, starting from a global OQL$_{I^3}$ query formulated by the user on the global schema. Using Description Logics techniques, the QM component can generate in an automatic way the translation of the generic OQL$_{I^3}$ query into different sub-query, one for each involved local source. The *query agents* are mobile agents in charge of bringing such sub-queries to the data source sites and there they interact with the local wrapper agents to carry out the queries; then they report the data result to the QM. To achieve the global answer, the QM has to merge each local sub-queries result into a unified data set. This process involves the solution of redundancy and reconciliation problems, due to the incomplete and overlapping information available on the local sources, i.e. *Object Fusion* [30].

For example, over the Global Virtual View build in the previous section we should “retrieve the car name and price for power levels present in every sources”, that is obtained by the following query:

```
Q: select V1.name, V1.power, V1.price,
   from Vehicle V1
   where 1 <= (select count(*)
                from  where Vehicle V2
                where V2.power = V1.power
                and   V2.source <> V1.source)
```

Processing the above global query would individuate all the local classes involved: `VW.Vehicle`, `VW.Model`, `VW.Motor`, `FIAT.car`, `FIAT.engine`.

The QM, by the query reformulation process [7], defines the following local queries (`QL1` expressed by SQL and `QL2` by XQuery [16]):

```
QL1: select M1.name, Motor.KW, M1.price
     from VW.Model M1, VW.Motor
     where M1.cod_m = Motor.cod_m

QL2: FOR $C in document("fiat.xml")//car
     RETURN
     <car>
           <name>$C/name</name>
           <price>$C/price</price>
           <kw>$C/engine/power_kw</kw>
     </car>
```

This process may be executed in parallel by two query agents that move to the two local sources. After performs the query, each query agents returns the data result to the MOMIS QM that fuses the two data-set obtaining the final result (in the example the fusion is obtained by join the data-set on the power attribute).

Following a more autonomous approach, for this class of queries (i.e., where a fusion of data derived by different local sources is necessary) the data process should be moved to the local systems to minimize the data transfer. This is performed by providing query agents with appropriate local service queries that obtain an intermediate result, which can be sent to other query agents to perform fusion at the local sites.

For example the following service queries using local query language are added:

```
QS1: select DISTINCT Motor.KW
   from VW.Motor

QS2: FOR $KW in distinct(document("fiat.xml")//car)
     RETURN
        <kw>$KW</kw>
```

These service queries are executed by the query agents to the local sources and the data results are exchanged with the other agents. In each single source the intermediate data sets are joined to the local queries to obtain the fused result (the join attributes are extracted from the mapping tables of which each query agent has a copy):

```
Q1: select DISTINCT QL1.name, QL1.KW,  QL1.price
    from QL1, QS2
    where QL1.KW = QS2.power_kw

Q2: select DISTINCT QL2.name, QL2.power_kw,  QL2.price
    from QL2, QS1
    where QL2.power_kw = QS1.KW
```

In this way, the union of `Q1` and `Q2` results are the same obtained by the first shown approach, while the data transfer amount with the MOMIS central site is drastically reduced, since only the request data are moved by the agent.

This example has shown the effective use and the related advantages of the exploitation of agents in our infrastructure for the coordination of data retrieval and management.

4 Related Work

As far as we know, there are few agent-based approaches in the area of information integration.

A significative is the MCC InfoSleuth(tm) [31, 21] Project 1. It is an agent-based system for information gathering and analysis tasks performed over networks of autonomous information sources. A key motivation of the InfoSleuth system is that real information gathering applications require long-running monitoring and integration of information at various levels of abstraction. To this end, 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 experience is the RETSINA multi-agent infrastructure for in-context information retrieval [33]. In particular the LARKS description language [34] is defined to realize the agent matchmaking process (both at syntactic and semantic level) by using several different filters: Context, Profile, Simlarity, Signature and Constraint matching. Differently from our approach, both InfoSleuth and RETSINA does not take advantage from the mobility feature of the agents, which we consider fundamental in a dynamic and uncertain environment such as the Internet.

In the area of heterogeneous information integration, many projects based on a mediator architecture have been developed. The mediator-based TSIMMIS project [17] follows a 'structural' approach and uses a self-describing model (OEM) to represent heterogeneous data sources, the MSL (Mediator Specification Language)rule to enforce source integration and pattern matching techniques to perform a predefined set of queries based on a query template. Differently from MOMIS proposal, in TSIMMIS only the predefined queries may be executed and for each source modification a manually mediator rules rewriting must be performed.

The GARLIC project [14] builds up on a complex wrapper architecture to describe the local sources with an OO language (GDL), and on the definition of Garlic Complex Objects to manually unify the local sources to define a global schema.
The SIMS project [2] proposes to create a global schema definition by exploiting the use of Description Logics (i.e., the LOOM language) for describing 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 [27] provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources. Given a query, the system will create a plan for answering the query using the underlying source descriptions. Algorithms to decide the useful information sources and to generate the query plan have been implemented. The integrated schema is defined mainly manually by the designer, while in our approach it is tool-supported.

Infomaster [22] provides integrated access to multiple distributed heterogenuos information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modeled by the user, and a core system that dynamically determines an efficient plan to answer the user's queries by using translation rules to harmonize possible heterogeneities across the sources. The main difference of these project w.r.t. MOMIS is the lack of a tool aid-support for the designer in the integration process.

5 Conclusions and Future Work

This paper has presented how a system for information integration can be improved by the exploitation of mobile agents. In particular, agents are useful in the management of the sources, which, in an open and dynamic scenario like the Internet, can be spread and can change in a uncontrolled way. The mobility feature permits to overcome the limitations of the traditional approaches of distributed systems.

With regard to future work, we sketch some research directions.

The first one relates to the dynamic integration of information sources. Currently, when a new source is added (or when is deleted), the MOMIS system has to be restarted. Our aim is to allow source integration at runtime, possibly by exploiting mobile agents that search for interesting new sources or check the request of an administrator for integrate her/his new source. This will permit to face the high degree of dynamism of the Internet.

Then, we are evaluating which further components of the system can be modeled as agents, and, in particular, which ones can take advantage from the mobility feature. The fact that some components may be mobile permits to deploy the whole system in a more flexible and adaptable way.

Finally, an area that is worth to be explored is the interaction between agents, which can occur in different ways and with different languages. On the one hand, complex languages for the knowledge exchange have been proposed [20]; on the other hand, mobility of agents promotes the adoption of simple and uncoupled coordination protocols [13].

References

1. Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
2. Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comprehensive analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):322–364, 1986.
4. I. Benetti, D.Beneventano, S.Bergamaschi, A. Corni, F. Guerra, and G. Malvezzi. Si-designer: a tool for intelligent integration of information. *International Conference on System Sciences (HICSS2001)*, January 2001. Available at http://www.dbgroup.unimo.it/prototipo/ paper/hicss2001.ps.gz.

5. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 611–614. Morgan Kaufmann, 2000.
6. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. In *Int. Conference on Data Engineering - ICDE97*, 1997. http://sparc20.dsi.unimo.it.
7. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogenous information sources. *Journal of Data and Knowledge Engineering*, 36(3):215–249, 2001.
8. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.
9. P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard, April 1996. Available at ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps.
10. G. Cabri, L. Leonardi, and F. Zambonelli. Agents for Information Retrieval: Issues of Mobility and Coordination. *Journal of Systems Architecture*, 46(15):1419–1433, December 2000.
11. G. Cabri, L. Leonardi, and F. Zambonelli. MARS: A Programmable Coordination Architecture for Mobile Agents. *IEEE Internet Computing*, 4(4):26–35, July/August 2000.
12. G. Cabri, L. Leonardi, and F. Zambonelli. Mobile-agent coordination models for internet applications. *IEEE Computer*, 33(2):82–89, February 2000.
13. G. Cabri, L. Leonardi, and F. Zambonelli. XML Dataspaces for the Coordination of Internet Agents. *Applied Artificial Intelligence*, 15(1):35–58, January 2001.
14. M.J. Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams, and E.L. Wimmers. Object exchange across heterogeneous information sources. Technical report, Stanford University, 1994.
15. R. G. G. Cattell, editor. *The Object Database Standard: ODMG 3.0.* Morgan Kaufmann Publishers, San Mateo, CA, 2000.
16. D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. Xquery: A query language for xml. In *W3C Working Draft 15 February 2001*, Feb 2001.
17. S. Chawathe, Garcia Molina, H., J. Hammer, K.Ireland, Y. Papakostantinou, J.Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ Conference, Tokyo, Japan*, 1994. ftp://db.stanford.edu /pub/chawathe/1994/ tsimmis-overview.ps.
18. DAML Joint Committee. Daml Project. Available at http://www.daml.org.
19. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, Lecture Notes In Artificial Intelligence. Springer-Verlag, 2000. To appear.
20. Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 14, pages 291–316. AAAI Press / The MIT Press, 1997.
21. J. Fowler, B. Perry, M. H. Nodine, and B. Bargmeyer. Agent-based semantic interoperability in infosleuth. *SIGMOD Record*, 28(1):60–67, 1999.
22. M. R. Genesereth, A. M. Keller, and O. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD Conference*, 1997.
23. J. Gilarranz, J. Gonzalo, and F. Verdejo. Using the eurowordnet multilingual semantic database. In *Proc. of AAAI-96 Spring Symposium Cross-Language Text and Speech Retrieval*, 1996.

24. R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.
25. Nicholas R. Jennings and Michael J. Wooldridge, editors. *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, Berlin, 1998.
26. Neeran M. Karnik and Anand R. Tripathi. Design issues in mobile-agent programming systems. *IEEE Concurrency*, 6(3):52–61, July/September 1998.
27. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB 1996*, pages 251–262, 1996.
28. S. E. Madnick. From vldb to vmldb (very many large data bases): Dealing with large-scale semantic heterogeneity. In *VLDB Int. Conf.*, pages 11–16, 1995.
29. A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
30. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.
31. B. Perry, M. Taylor, and A. Unruh. Information aggregation and agent interaction patterns in infosleuth(tm). In *Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems*, 1998.
32. M.T. Roth and P. Scharz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, 1997.
33. K. Sykara. In-context information management truough adaptative collaboration of intelligent agents. In *Intelligent Information Agents*, pages 78–99. M. Klusch Ed. - Springer, 1999.
34. K. Sykara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Records*, 28(1), March 1999.