MIKS: an agent framework supporting information access and integration *

Domenico Beneventano[1,2], Sonia Bergamaschi[1,2],
Gionata Gelati[1], Francesco Guerra[1], and Maurizio Vincini[1]

[1] Dipartimento Ingegneria dell'Informazione, via Vignolese 905, 41100 Modena Italy
[2] CSITE-CNR, viale Risorgimento 2, 40136 Bologna, Italy
{domenico.beneventano,sonia.bergamaschi,maurizio.vincini}@unimo.it
{gelati.gionata,guerra.francesco}@unimo.it

Abstract. 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, not homogeneous and not always semantically consistent. 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. MOMIS (Mediator EnvirOnment for Multiple Information Sources) is a system that has been conceived as a pool of tools to provide an integrated access to heterogeneous information stored in traditional databases (for example relational, object oriented databases) or in file systems, as well as in semi-structured data sources (XML-file).
This proposal has been implemented within the MIKS (Mediator agent for Integration of Knowledge Sources) system and it is completely described in this paper.

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, not homogeneous and not always semantically consistent. 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,

* This work is partially supported by MIUR co-funded projects D2I and 'Software Agents and e-Commerce'

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 [4] 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 [48]. Furthermore, 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. *MOMIS* [8, 16] (Mediator EnvirOnment for Multiple Information Sources) is a system that has been conceived as a pool of tools to provide an integrated access to heterogeneous information stored in traditional databases (for example relational, object oriented databases) or in file systems, as well as in semi-structured data sources (XML-file). *MOMIS* focuses on capturing and reasoning about semantic aspects of schema descriptions of information sources for supporting integration and query optimization. This proposal has been implemented within the *MIKS* (Mediator agent for Integration of Knowledge Sources) system and it is completely described in this paper.

Mobile agents can significantly improve the design and the development of Internet applications thanks to their characteristics. The agency feature [39] 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 [41] 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 [20].

MIKS is an agent framework for information integration that deals with the integration and query of multiple, heterogeneous information sources, containing structured and semi-structured data. This framework is a support system for semi-automatic integration of heterogeneous sources schema (relational, object, XML and semi-structured sources); it carries out integration following a semantic approach which uses Description logics-based techniques, clustering techniques and an ODM-ODMG [24] extended model to represent extracted and integrated information, ODM_{I^3}.

The *MIKS* system can be defined as an agent middleware system that integrates data belonging to different and potentially heterogeneous sources into a

global virtual view and offers support for the execution of queries over the global virtual schema [15]. Middleware systems dealing in some way with a set of data sources commonly fall back on wrapper components or simply wrappers [54]. Wrappers components encapsulate existing legacy data sources and give a more presentable and understandable format according to some preferred common data model.

The outline of the paper is the following. Section 2 presents the basics related to the approach we have chosen in tackling the integration of heterogeneous data sources, section 3 reports the *MOMIS* system. Section 4 introduces the *MIKS* system, illustrating the role of the agents in a framework supporting information access and integration. Finally, section 6 discusses the related work in the area of intelligent information agents.

2 A feasible approach to intelligent information integration architectures

In this section we introduce the basics related to the approach we have chosen in tackling the integration of heterogeneous data sources. In the first sub-section we present the reference architecture and in the second sub-section we step through the integration process.

2.1 System Architecture

Like in other integration projects ([2, 57]), we have chosen to pursue a "semantic approach" to information integration, i.e. we represent the content of information sources by means of conceptual schemas (or in other terms metadata). This allows to process data not only from a syntactical point of view but also according to their meaning in order to infer extensional and intensional relationships among them. Given the goal of our system, our reference architecture has been the I^3 architecture as specified in [38] for the Intelligent Integration of Information.

Figure 1 shows the five fundamental families of I^3 Services and the primary ways in which they interact. In particular, two salient axis are definable to emphasize the different roles of the I^3 services. The vertical axis, that spans the families (1), (3), (5), is focused on the flow and manipulation of information from raw Information Sources up to the Coordination Services. Then, an horizontal axis connects the Management family and the Coordination family. This axis emphasizes a critical aspect of the I^3 Reference Architecture, i.e. the role of the Management family in order to locate useful information sources and to exploit the local data structures.

Our system mainly exploits services belonging to the vertical axis: in particular the family (3) has been completely investigated. In previous work, we faced the issues related to the semantic integration and transformation, developing and implementing a methodology to integrate schema of heterogeneous information sources. Our research has highlighted that besides the functional elements belonging to the reference system architecture, a set of functional extensions

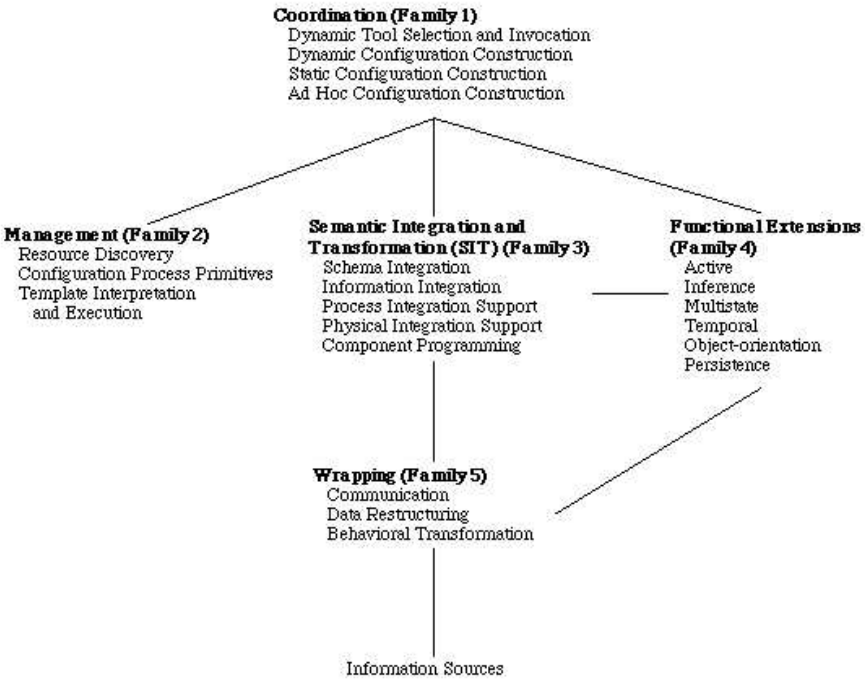

Fig. 1. Reference System Architecture

(family 4 in the architecture) is required in order to support designers during the integration process, especially when delaying with both semi-structured and structured sources. Thus, our approach has lead us to the introduction of the following services: a Common Thesaurus which has the role of a shared ontology of the source and reasoning capabilities based on Description Logics.

More precisely, the Common Thesaurus builds a set of intra and inter-schema intensional and extensional relationships, describing inter-schema knowledge about classes and attributes of sources schemas. Further, designers are left free to supply any further domain knowledge that can help refine the integration process. The Common Thesaurus provides a reference on which to base the identification of classes candidate to integration and subsequent derivation of their global representation.

As one of the goals of our integration system is revising and validating the various kinds of knowledge used for the integration, we have combined within the architecture reasoning capabilities of Description Logics with affinity-based clustering techniques.

In the following, we will consider the functional elements related to the Coordination family and propose a Multi-Agent System where coordinations take place in order to accomplish the required integration and querying functionalities.

2.2 The Integration process

The overall information integration process we have assumed for our purposes 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. In the Common Thesaurus, we express inter-schema knowledge in form of terminological and extensional relationships (synonymy, hypernymy and relationship) between classes and/or attribute names;
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.**
   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.

2.3 Query processing

A data integration system, based on conventional wrapper/mediator architectures, usually allows the user to pose a query and receive a unified answer without the need of: locating the sources relevant to the query, interacting with each source in isolation and combining the data coming from the different sources. In our framework, the user application interacts with the system to query the Global Schema by using the OQL_I^3 language. Using the mapping between local and global attributes, the Query Manager generates in an automatic way the reformulation/optimization of the generic OQL_I^3 query into different subqueries, one for each involved local source (see [10]). To achieve the mediated query result, the Query Manager has to assemble each local sub-query result into a unified data set.

In a mediator architecture, the availability of several heterogeneous sources adds many novel issues to the query processing and optimization problem. From a theoretical point of view, solving a user (mediated) query, i.e. giving a single unified answer w.r.t. multiple sources, implies to face two main problems: *query reformulation/optimization* [31, 47, 30, 32, 56, 40, 44] and *object fusion* [53, 63].

Semantic Query Optimization In a mediator architecture the query manager usually relies on the availability of a global schema, the source schemata and a mapping between the two. On the other hand, the heterogeneity of information sources to be integrated often entails significant overlap or redundancy among them. Exploiting such an extensional knowledge is an important task in the query planning phase since most of the execution costs concerns the cost for querying remote sources, because of the high connection overhead, long computation time, financial charges and temporary unavailability. In [10] we have discussed the introduction of extensional knowledge as an additional knowledge allowing semantic optimization during the query planning phase.

In this context, besides incoherence detection and factor removal, the main achievements of semantic query optimization are related to the minimization of the number of sources to be accessed and the maximization of the selectivity of the query sent to the sources.

To illustrate the problem, let us consider a mediator integrating two sources. Suppose that the first source provides class `Student(name,email,year,tax)` and that the second source has `Professor(name,email,dept)`. Let us assume that the outcome of the integration phase is the mediated global schema with the global class `UniversityPerson(name,email,year,tax,dept)`. Let us consider the following query:

```
Q:  select email
    from University_Person
    where year = 2002
    and (dept ='cs'  or tax < 200)
```

Our semantics of a global class is based on the following hypothesis, called semantic homogeneity: the objects of `UniversityPerson` are objects of `Student` and/or of `Professor`; objects instantiated both in `Student` and `Professor` are individuate by considering the name attribute value and these objects have the same value for the common attributes. Thus, intuitively, the above query has to be reformulated on the basis of the local classes, as follows:

```
QS: select email,name,tax
     from Student
     where year = 2002

QP:  select name
      from Professor
      where dept = 'cs'
```

and then:

```
Q:   select email
     from QS
     where tax < 200
     union
     select email
     from QS, QP join on (QS.name=QP.name)
     where year = 2002
```

Now, let us suppose that the integration designer introduces the extensional knowledge that the local classes `Student` and `Professor` are disjoint. Then, in the above reformulation the answer of QS, QP join on `QS.name=QP.name` is empty and thus Q has to be reformulated only on the local class Student, as follows:

```
Q: select email
     from Student
     where year = 2002 and tax < 200
```

This example shows the main achievements of semantic query optimization of our method: (1) the minimization of the number of sources to be accessed and (2) the maximization of the selectivity of the query sent to the sources.

2.4 Managing extensional knowledge

A fundamental aspect of semantic query optimization is the management of the extensional knowledge. The main issue is that a set of extensional assertions

which is initially verified by local sources can be violated if data are modified in one or more sources over time. Checking the consistency of the data against the extensional assertions is a very complex and challenging point. In [27] a framework to monitor and enforce distributed integrity constraints in loosely coupled heterogeneous information systems is presented. In our context, as a mediator is not the owner of the data stored in the local classes but it only provides a virtual view, we are interested only in the *verification* of integrity constraints. Nonetheless finding a solution still remains a complex activity. Thus, the main goal has to be that of reducing the transfer of data among remote heterogeneous sources. A useful optimization in a mediator environment is to be able to verify global extensional assertions by only accessing local data. Reference [36] describes a method for checking distributed constraints at a single source whenever possible. Given a global constraint and some data to be inserted in a local source, a local condition is produced in order to verify the consistency of local data. The consistency of local data is proven if (a) the local condition is (locally) satisfied and (b) before the new data were inserted, the global constraint was satisfied. this optimises the verification process as no remote data have to be accessed.

3 The *MOMIS* project

Given the set of functionalities and how they have to work, a system can be designed following diverse approaches. A classical object-oriented implementation is feasible [42]. This approach is particularly suited to achieve client-server architectures. This way, the server acts as a centralised facility where all the computation takes place, while clients can be envisaged as lighter desktop applications and user interfaces. The server could comprise of more machines: the objects that provide the services on the server side can be distributed, typically using the CORBA [35] standard framework. Anyway, we can still maintain the server as a logically unified facility although its services are distributed over a local cluster of servers. This configuration is useful when the system is meant to operate in a limited and close search domain, i.e. search domains that are well-known as far the type and the number of sources to be integrated. A typical scenario of this sort could be represented by a small enterprise whose main goal is managing the few heterogeneous data sources belonging to the local information system. Users of the systems are usually the few analysts charged within the organisation of data monitoring and retrieval.

Within the *MOMIS* (Mediator envirOnment for Multiple Information Sources) project [17, 6, 16], we ourself have experienced the design and implementation of an object-oriented client-server integration system, that follows the architecture above presented. The *MOMIS* system has obviously been conceived to provide an integrated access to heterogeneous information stored in traditional databases (e.g., relational, object oriented) or file systems, as well as in semi-structured sources (*XML* files in particular). Figure 2 shows the architecture of the *MOMIS* system, based on the I^3 schema of Figure 1 : at the bottom layer

we have the schema of information sources, while the layers above provide the semantic integration and the coordination management support.

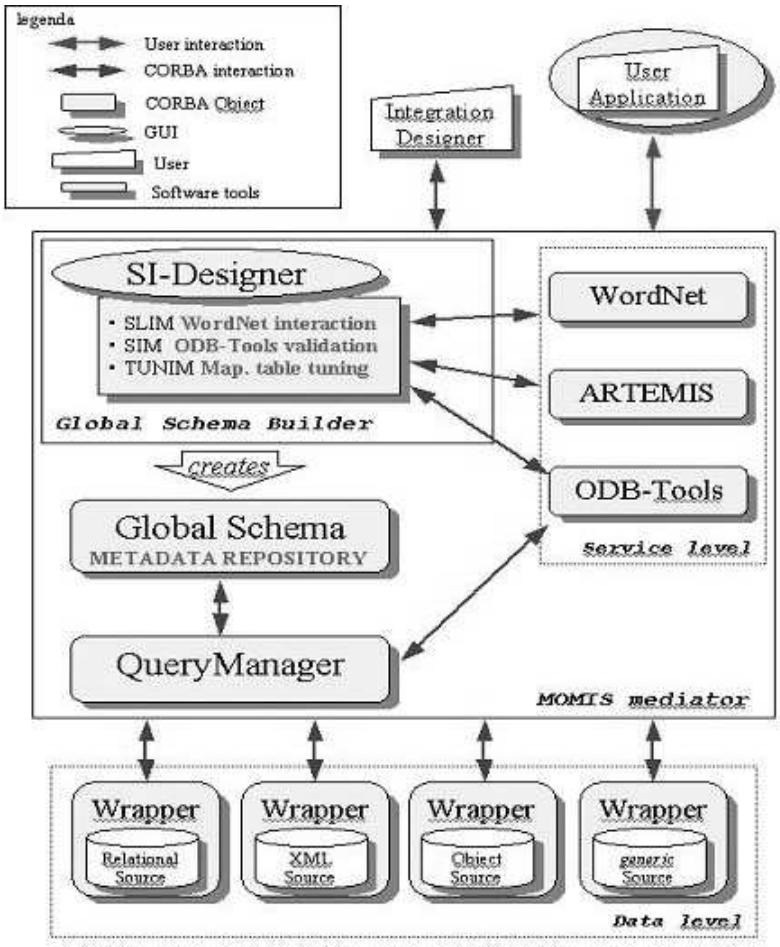

Fig. 2. The architecture of the *MOMIS* system

As depicted in Figure 2, the system is composed by the following functional elements that communicates using the *CORBA* standard:

– Wrappers: The wrappers in *MOMIS* are the access point for the data sources. This implies that independently from the type of the underlying source (relational, object, structured and so forth), data must be presented in a standard way. Wrappers are *CORBA* objects connected to some sources and are able to describe the source structure using the ODL_i^3 language. Further, wrappers supply a way to query the source using the OQL_i^3 language;

- Mediator: It is composed of two modules: the *Global Schema Builder* (*GSB*) and the *Query Manager* (*QM*). The *GSB* module processes and integrates ODL_{I^3} descriptions received from wrappers to derive the integrated representation of the information sources. The *GSB* is the main *CORBA* object to access the *MOMIS* integration services. The *GSB* is composed by two modules:
  - *SIM* (Source Integrator Module): extracts intra-schema relationships starting from a relational, object and semi-structured source. Moreover this module performs the semantic validation of relationships and infers new relationships by exploiting *ODB-Tools* capabilities.
  - *SLIM* (Sources Lexical Integrator Module) extracts inter-schema relationships between names and attributes of ODL_{I^3} classes of different sources, exploiting the *WordNet* lexical system.

  All information about integration are then saved in the Global Schema which is defined as a *CORBA* object. A Global Schema object contains all information for querying the resulting global schema by a query manager object. The *QM* module performs query processing and optimization. In particular, it generates the OQL_{I^3} queries for wrappers, 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 global OQL_{I^3} query into different sub-queries, one for each involved local source. The *QM* is also responsible for synthesising the results of the sent sub-queries into one global answer to the original global query;
- The *ODB Tools* Engine ([12]), a tool based on the *OLCD* Description Logics ([9, 18] which performs schema validation for the generation of the *CommonThesaurus* and query optimization ([11];
- The *ARTEMIS Tool Environment*, a tool based on affinity and clustering [22, 23], which performs classification of ODL_{I^3} classes for the synthesis of global classes.
- The *SI-Designer* module, which provides the designer with a graphical interface to interact with *SIM*, *SLIM* and *ARTEMIS* modules showing the extracted relationships and helping her/him in the Common Thesaurus construction. This way, the integration develops as a semi-automatic process where the designer can evaluate the result of the various steps and eventually try to modify the result according to his/her understanding. Once the Common Thesaurus, has been built, *SI-Designer* uses the *ARTEMIS* module to evaluate a disjoint set of structural similar classes. *SI-Designer* automatically generates a set of global attributes for each global class and a mapping table which maps each global attribute into the local attributes of the classes in the cluster.

  For a detailed description of the mappings selection and of the tool *SI-Designer* which assist the designer in this integration phase see [5].

Finally, in *MOMIS* the extensional knowledge is provided by the integration designer as a set of assertions: containment, equivalence and disjunction among local classes of the sources involved in the integration. The basic assumption

is that these designer-supplied assertions are always verified over time. In the actual implementation, there is no verification tool to monitor the state of the sources against the extensional assertions.

4 The role of the agents: the *MIKS* system

When the application horizon gets wider, the search has to be extended to a larger and more distributed amount of data, heterogeneity related to data structures and semantics increases, the number of users can become much more sizeable and services based on a client-server architecture may not represent the most suitable and comprehensive solution. The system has to meet requirements against flexibility in terms of system configuration and user needs, pro-activeness in terms of responses to a changing environment, scalability in the number of users and reliability in terms of coping with an unstable environment. An approach that tackles these issues and that has attracted the attention of the research community especially during the last decade is that based on software agent technology [19] and multi-agent systems *(MASs)*[62]. The agent paradigm [59] proposes to design and implement systems where agents act autonomously while they can still aim at cooperating with other agents to achieve a complex task. *MASs* are those where agents are organised in societies where communication, interaction and coordination are possible. In our work we are concerned with intelligent information agents [43]. They are mainly characterised as holding intelligence (they manipulate information) and mobility (they can move through the network). The *MIKS* system hence comprises of a society of agents providing advanced information management functionalities. The advantage of the use in the *MIKS* infrastructure of intelligent and mobile software agents for the autonomous management and coordination of the integration and query processes have been introduced in [14]. Related work on the use of agents in information management systems (such as the Infosleuth project) is reported in section 6.

In the *MIKS* system, the exploitation of intelligent information agents (agents for short) improves the flexibility, as a number of system configurations are possible. As each component (i.e. agent) can be placed at whatever available host or data source and still provide the service it has been designed for, what in a client-server approach are the server side functionalities can be distributed and further moved while the system is running. On the client side, different solutions are possible in that agents allow for a number of devices to use the *MIKS* system, from desktop computers to PDAs to mobile phones. Agents can show to users services and interfaces according to the particular features of the device in use and deliver contents while users are moving wearing their portable devices.

Decomposing the system into independent and autonomous entities allow to set up configurations that respond to diverse level of granularity. According to the specific application scenario, the designer can choose whether to consider each component as a stand-alone entity or to group some components to create macro-entities whose components are joint as a whole. Throughout the design

phase, we have chosen to define agents according to the step they come into play (when) and the service they provide (what) within the integration process.

Notice that, following [43], mediators agents are characterised by three capabilities:

1. translating among diverse ontological domain to build a uniform semantic view of a given domain of interest
2. decomposing and executing complex queries
3. fusing partial answers.

We can distinguish between pure integration capabilities (1) and querying capabilities (2 and 3). We have therefore split the system into two parts. The first is meant to provide the functionalities for supporting the integration process, while the second supports the querying phase. This distinction we have taken when designing the agent-based *MIKS*system. The resulting architecture comprises of two multi-agent systems (the first supporting the integration process and the other supporting the querying phase). The two multi-agent system are meant to be interoperable.

This has produced the side effect of splitting the functionalities of components (see Figure 1) according to the phase they participate in. For instance, wrappers are traditionally meant to support both the translation of the data source schema into a common data model and the execution of queries. In building the *MIKS* wrapper architecture there have been some driving goals. First, wrappers should be able to evolve over time. Wrappers should have the dynamic capability of adding new functionalities to their core behaviour. Secondly, wrappers should be flexible enough to be able to wrap sources which expose different services and capabilities, such as query processing capabilities. Thirdly, wrappers should have strong interaction capabilities with components requesting either information about the source or information contained in the source. Agents hold the properties such as pro-activeness and the ability to adjust their behaviour according to the environment that we find well-matched for designing and implementing wrappers. In our framework, these functionalities have been kept separated (namely the *Translator agent* and the *Query Support Agent*) because they come into play in realising two different services of the *MIKS* system. Defining the *MIKS* wrappers, we have found agent technology particularly useful. It defines a very flexible way of managing distributed components, from creation to migration to replication, that help shape the system configuration according to the particular operative scenario. The strong support for interoperability and conversation inherently distinguishing agents has represented an advantage as far as interactions and information exchange among the system components is concerned.

4.1 A Multi-Agent System for supporting the *MIKS* integration process

The MAS we have designed for integration purposes includes agents that support all of the four phases we have presented in section 2.2. Agents carry out activities

for manipulating data to create information at higher levels of abstraction. Figure 3 depicts how agents are organised within the MAS.

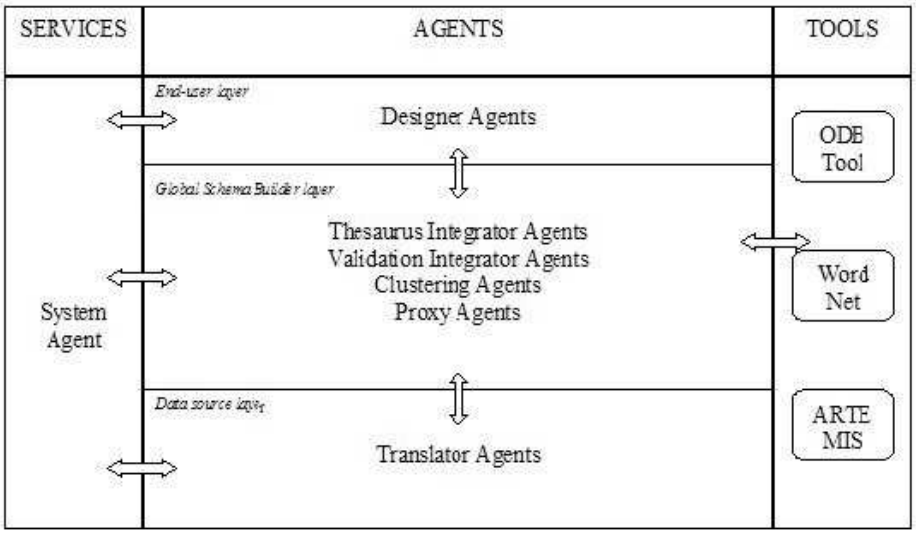

Fig. 3. Classification of agents for supporting the integration process

The arrows highlight that the information can flow between the agents of different layers at either end of the arrows.

Starting from the left side of Figure 3, we find the *System Agent (SA)*. It performs administrative tasks like system configuration monitoring and agent life-cycle management.

As the distribution of agents has the primary aim of distributing the workload to the *MIKS* system, the *SA* acts as the system monitor. It helps estimate the system status and workload and it is the one which decides when and where agent should migrate. This in turn realises the workload balancing mechanism. The *SA* manages the life-cycle of agents. It makes provision of an environment which is meant to host agents (code, data and status) and permits them the execution of their algorithms. The *SA* and more in general the agents, have been built using the JADE environment [45], a FIPA-compliant development tool [33].

The middle and right-side column show the agents and tools required for the integration process. Agents are grouped along three layers.

Rising up from the bottom of the middle column, in the *Data Source layer* we find the *Translator Agents (TAs)*. A *TA* acts during the phase in which

the source is recognised and its content has to be expressed using the ODL_I^3 language. The *TAs* have the following functionalities inside the *MIKS* system:

1. they can inherently adapt to a variety of data sources (relational DBMS, object-oriented DBMS, structured and unstructured documents)
2. they build the ODL_{I^3} description of the underlying data source(s) and keep it up-to-date according to the committed changes. Whenever this happens, *TAs* communicate the changes to the agents belonging to the upper level, the *Global Schema Builder layer*
3. they provide a description of the services and query capabilities of the source and constantly report their status to the upper layer agents. Knowing the query capabilities of the sources is essential during the translation of the global query into the set of local queries in order to draw a query plan that is actually executable. Some interesting proposals facing the problem of querying information sources which provide different levels of query capabilities have been introduced in [25, 37, 57]. The aim is pruning efficiently out information sources that are unsuitable for a given query and generating executable query plans.

In the *Global Schema Builder layer* are grouped the agents that support the integration process. *Proxy agents* are information collectors. They acquire the local schemata created by the *TAs*. This knowledge base will be manipulated and enriched during the integration activities by the other *Global Schema Builder layer* agents and maintained in the corresponding *Proxy agents*.

After having acquired the set of local schemata, *Thesaurus Integrator Agents* (TIAs) and *Validation Integrator Agents* (VIAs) carry out the reasoning and inference activities required to semantically integrate data. *TIAs* are charged of extracting intensional intra/inter schema relationships. They have been subdivided into two types:

- *TIAs* that separately analyse the local ODL_{I^3} schemata in order to extract terminological and extensional relationships holding at intra-schema level
- *TIAs* that analyse the set or subsets of local ODL_{I^3} schemata in order to extract terminological relationships holding at inter-schema level.

VIAs are charged of interacting with the ODB-Tools to infer new relationships that hold at intra-schema level. Further, *VIAs* validate the whole set of relationships that have been discovered so far.

Together with designer-supplied relationships (as we will see when describing *Designer Agents*), these are used to build the *Common Thesaurus*, which represents the ontology of the given domain of integration. This completes the first two steps of the integration process. *Clustering Agents* generate the set of structural similar classes (*clusters*) and the corresponding global classes. This leads to the generation of global attributes and a mapping-table.

Notice that a common feature of the agents belonging to the *Global Schema Builder layer* is the interaction with the *MIKS* system tools. These are either knowledge bases (as the *WordNet* data base) or already existing applications

(as *ARTEMIS* and *ODB-Tools*) developed in other research projects. All of the tools have sufficiently sophisticated interfaces to allow interaction with other applications. Thus, it is easy to make them interoperate with agents. This can be done by agentifying these applications (agents that are expressively being designed for exchanging data and calling functions of the tool) or by exposing the functionalities of the tools as web services.

At the end-user level, *Designer Agents (DAs)* are available. Their main functionality is providing designers a graphical user interface towards the *MIKS* system, its configuration and the data it integrates. They are much like the *MOMIS* SI-Designer module that provides the designer with a graphical user interface that shows the *Global Virtual View (GVV)*. Details can be found in [5, 8]. The main difference is that *DAs* do not directly interact with the *MIKS* tools, but only with agents. A *DA* collects this information by interacting on a regular basis or on-demand with the agents presented so far. The system configuration is retrieved from the *SA* while an overview on sources, local schema, terminological and extensional relationships (inter- and intra-schema) are retrieved from the underlying layers. Further, *DAs* allow designers interacting with the other agents (and indirectly with the *MIKS* system tools), thus enabling control over the integration process (for instance, choosing the sources to be integrated and selecting the "best" clustering configuration among the proposed alternatives).

4.2 A Multi-Agent System for supporting global query execution

The Global Schema gives users an integrated view over data that were previously scattered over different places and applications and had to be accessed separately both from a logical and physical viewpoints. This is a very powerful service given the initial heterogeneous information environment. Nevertheless, a comprehensive attitude towards information management includes not only displaying the whole range of data, but also a way of submitting queries in order to select particular set of data. This is very useful if we want to restrict the focus of our analysis to what is really important for our purposes. The *MIKS* system allows users to submit queries over the Global Schema (global queries). Similarly to other semantic approaches, the querying phase consists of three steps:

1. semantic optimization
2. query plan execution
3. fusion of local, partial answers.

We refer the reader to [10] for more details about the techniques deployed to realise these three steps. We have designed a MAS for supporting the whole phase of global query solution. Agents perform activities for manipulating global queries to create queries at lower level of abstraction (local queries) that are hence executable on data sources. Local answers have then to be synthesised into a global answer. Notice while the integration process is essentially a one-way bottom-up information flow starting from the source contents and ending up with the generation of a *GVV*, the querying phase is a two-way process: top-down

when users submit queries over the GVV and bottom-up when local answer are made available and have to be merged to compose the global answer. Our MAS reflects the nature of the querying process. Figure 4 illustrates the organisation of agents. The arrows highlight that the information can flow between the agents

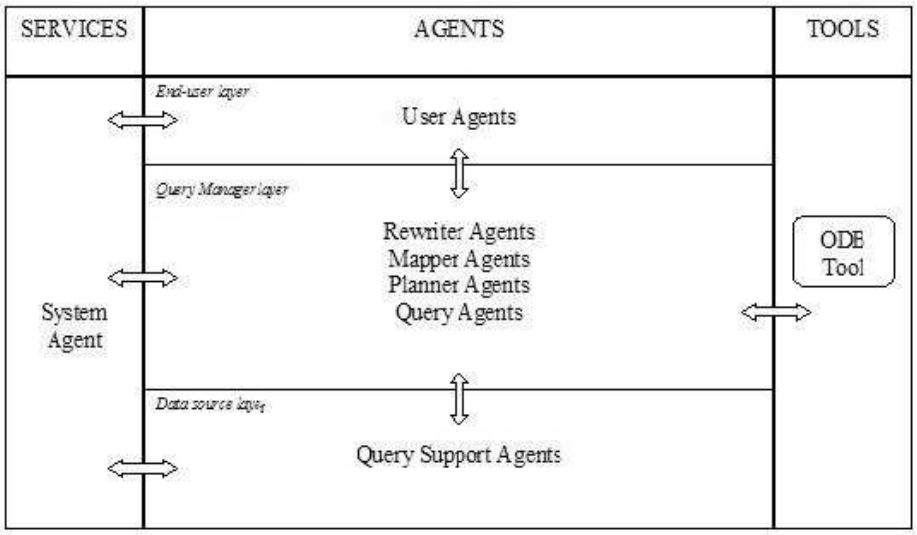

Fig. 4. Classification of agents for supporting the querying phase

of different layers at either end of the arrows. The right-side and left-side columns are the same we have just described above for the MAS supporting information integration. We will focus on the middle column. This time we will proceed from the top layer down.

In the end-user layer we find *User Agents* (UAs). This reflects the shift in the type of system users: information integration requires interaction with designers, query processing is expressively made available so that users can submit actually access data they are interested in.

In the setting of distributed services available on the network, users do not usually get access to the desired services directly at the facility that provides them. They use applications running on hosts they are using. Besides the traditional wired way of getting connected to local or wide networks or the Internet, in recent years the usage of wireless devices has become more and more recurrent. These kinds of mobile devices are characterised by strong limitations in terms of battery autonomy, processing power and bandwidth if compared to desktop

and personal computers. These limitations have to be taken into account when interfacing the *MIKS* system to mobile users. Falling back on client-side applications could not be the most effective solution and sometimes it could be even impracticable to install them (especially when needs arise on the fly).

User-side applications offer an interface to some underlying capabilities that ultimately allow the access to services residing on some remote hosts on the network. The wide acceptance of the World Wide Web (simply Web) [28] as main application for information delivering makes web browsers a very appealing means for supplying user interfaces whatever the underlying services are. Often a simple web browser is not sufficient to enable users the usage of more complex services. An example is the provision of web-based database connectivity that requires the usage of *JDBC* drivers [50]. A detailed discussion of this topic can be found in [55]. In general, for web-based services specific capabilities have to be added as client side applications and web browsers have to get hold of them on the fly. This is the reason modern web browsers are Java enabled. There seems to be two major approaches to the provision of complex services on the Web. The first foresees code downloading to user hosts in order to enable the access to desired services. It is the case of *Java applets* ([49]) . The second creates a process whenever a user requests a service. The process serves that specific request and then terminate. It is the case of *Java servlets* ([51]). We will not take into consideration this second choice as it refers to traditional client-server architectures. *Java applets* can be seen as a first step towards agent technology: they move to the user host on-demand and execute some task on his/her behalf. For instance, current approaches to distributed database access based on *applets* have become increasingly popular in the last decade. As Papastavrou et al. show, a drawback of the use of *applet*-based service interfaces in this setting is that not only code has to be downloaded (that is pretty much comprehensible and inescapable) but at least an additional *JDBC* driver has to be downloaded and initiated. This process results in heavy procedures that could be not feasible in the case of devices with bounded resources (especially low memory space and low bandwidth connectivity). Agent technology is a way to overcome these limitations, adapting the fruition of the provided services to the particular features and capabilities of the worn device. Defining *UAs* means in turn defining a new kind of lightweight interfaces to remote services. *UAs* are responsible for collecting user queries, user preferences and device features in order to build user profiles. User queries are collected as OQL_{I^3} statements. As future work, we are going to provide users with a more advanced and powerful interface, like the one proposed in [52]. User profiles serve in turn to know which priorities have to be deemed when assessing the relevance of the information and how they can be transmitted and displayed to end users. Further, exploiting user agents has another important facet: users have to keep connections up only for a limited amount of time, i.e. the time required to send the agent code and data and the time required to collect results. Altogether, the user agent acts as a filter against the data they are retrieved by the *MIKS* system. The other modules of the *MIKS* system cannot execute this functionality as they are meant to provide

the overall outcome of the integration and querying processes (GVV and global answer). The filtering has to be done according to user preferences and device capabilities. User agents endow with a realistic mechanism to pledge a certain quality of service.

In the *Query Manager layer* are grouped the agents that carry out global query decomposition and partial answer merging. *Rewriter Agents (RAs)* operate on the query by exploiting the semantic optimisation techniques [7] [12] [58] supported by ODB-Tools [9, 29, 13] in order to reduce the query access plan cost. The query is rewritten incorporating any possible restriction which is not present in the global query but is logically implied by the Global Schema (class descriptions and integrity rules).

Mapper Agents (MAs) express the rewritten global query in terms of local schemas. Thus, a set of sub-queries for the local information sources is formulated. To this end, *MAs* dialogue with *Proxy Agents* that hold the knowledge about mapping table, global and local schema. In order to obtain each local query, the mediator checks and translates every predicate in the where clause.

Planner Agents (PAs) are charged to take the set (or subsets) of local queries and produce the executable query plan. The goal of *PA* is to establish how much parallelism and workload distribution is possible. Considering that queries are assigned to *Query Agents (QAs)* that move to local sources, creating a plan means trying to balance different factors:

- how many queries have to be assigned to each single *QA*
- which sources and in which order each *QA* has to follow in order to solve the assigned queries or to fuse partial results.

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. This allow for decentralisation of the computational workload due to collecting local answers and fuse them into the final global answer to be carried to the user. Future work will take into consideration more advanced techniques as reported for instance in [46, 1].

QAs move to local sources where they pass the execution of one or more queries to *Query Support Agents (QSA)*. A number of advantages are implied by moving to local sources. 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, users 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 as soon as the connection is available again. Finally,

this fits well mobile computing, where mobile devices (which can host user applications, agents and/or sources) do not have permanent connections with the fixed network infrastructure.

QSA afford translation services between the OQL_I^3 language and the native query language of the data source. This step is required to make queries executable by local information management system.

When the local answer is available, the corresponding *QA* has to map these data (whose structure follows the local schema) to the global schema. For doing this, the *QA* interacts with *Proxy Agents* to know the set of mapping rules related to the source. For instance, given that the attribute `name` in the Global Schema maps to the attributes `lastname` and `firstname` in the local source, the *QSA* has to put together the values of the two attributes in order to obtain the value for the global attribute `name`.

4.3 Further agent roles

Besides supporting the integration process and the querying phase, agents are suitable within the *MIKS* system for a number of complementary and precious tasks. In particular, we highlight here the extensional assertions on the data sources and the searching for new potential data sources.

Agents managing extensional knowledge As we have already stated, in the implementation of *MOMIS*, there is no verification tool to monitor the truth of the extensional assertions on the data sources. We have also seen that methods to check and enforce integrity constraints over a distributed set of information sources have been defined [27, 36]. We claim agent technology can be efficiently deployed to develop a framework for constraint verification in a mediator environment. Intuitively we can define *Check Extensional Assertion Agents (CEAAs)* for each data source. In order to carry out its task, a *CEAA* should know the assertions concerning the classes of the source and the operations on the source data (held by *Proxy Agents*) that can potentially lead to a violation of the extensional assertions. Whenever such an operation occurs, the *CEAA* communicate and coordinate with the other *CEAAs* to check the validity of the extensional assertions. Intuitively, such activity requires some transfer of data among data sources. Our goal is to integrate existing approaches as [27, 36] within our framework. We leave these new refinements of our framework as future work.

Searching for new data sources The set of remote sources to be integrated into the *MIKS* global view can be made up by sources autonomously deciding to grant the system their knowledge. On the other hand, at any moment a need to increase the number of the sources may arise in order to enhance the quantity and quality of available data. In such a case, the system has to search for new potentially interesting sources in a certain domain (e.g. Internet). The search may exploit in this case intelligent and mobile agents. For doing so, the system has to form the list of sources to be visited. This could be done by means of a

domain search according to the ontology contained into the *Common Thesaurus* and held by *Proxy Agents*. Then, the *SA* creates and coordinates a set of agents to be deployed. The *SA* allots the addresses of the list among some *Hunter Agents (HAs)*. Different policies can be enforced to control the activity of the *HAs* and to keep track of the state of the search. Once arrived at the assigned site, a *HA* has to introduce itself, clarify the goal of its visit and negotiate the availability of the source to grant access to a part or the whole of its content. If the result of the negotiation phase is positive, a *TA* has to be sent to the source.

5 Agent interaction patterns

As we have seen, diverse types of agents populate the two MASs of the *MIKS* system. During the design phase, the main functionalities of the system have been decomposed into a number of smaller services assigned to agents. Agents can move, thus relocating the services they provide. Figure 5 and 6 show the two extreme configurations the system can operate:

1. the former depicts how a client-server-like configuration is realisable,
2. the latter depicts a fully distributed architecture.

Arrows depict the most important interactions that occur among the system components.

The *MIKS* agents interact in order to achieve the overall goals of the *MIKS* system. Many interaction patterns are feasible as information has to flow from one agent to another. This can happen among agents either belonging to the same MAS or belonging to two different MAS. In the end, splitting the system into two MASs has served the purpose of keeping the architecture design clean avoiding waste of resources or overlapping in tasks among the agents.

In this section we will present some possible interaction patterns involving the agents we have presented so far. The purpose it to convey the reader the dynamics of agent interactions that can happen within the *MIKS* agent architecture. For the sake of clarity, we will refer to agents specifying their full name.

In a possible real scenario (Fig. 7), the *System Agent* spawns the agents required for the particular activities to be carried out.

At the beginning of the integration process, it interacts with the *Designer Agent* which reports which source have to be integrated. The list of interesting data sources can be available or filled with the suggestions of *Hunter Agents*. Then, the *System Agent* spawns *Translator Agents* in order to wrap the desired data sources. Arrows are bi-directional to mean that agents engage in dialogue in order to exchange information and that interactions may happen not only during initialisation but whenever need arises and throughout the whole system life.

During the integration process there are two main interaction patterns:

1. agents communicating with *Proxy agents* for retrieving data to be manipulated and storing enriched information

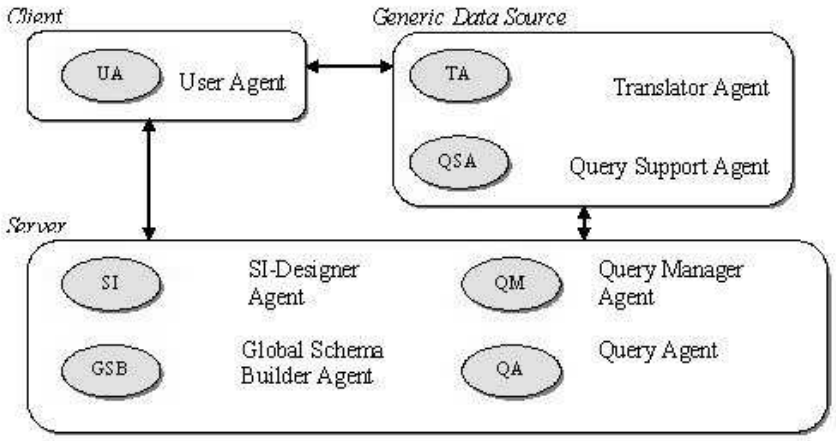

Fig. 5. The *MIKS* agents organised into a client/server configuration

2. *Designer Agent* communicating with agents belonging to the *Global Schema Builder layer* in order to interact during the various steps of the integration process.

Figure 8 show the complete set of interactions. Notice the *User Agents* can request and obtain from *Proxy Agents* the *Global Virtual View*, while *Hunter Agents* can update the reference ontology according to which the search has to be carried out.

The query solving phase foresees a more structure interaction pattern. Global queries are posed by users and then manipulated so as to be executable at the *Data Source layer*. The order of interactions follows the sequence of step required to decompose and solve the query (Figure 9):

1. *User Agents* directly transmit the submitted global queries to the *Rewriting Agents*
2. *Rewriting Agents* dialogue with *Proxy Agents* to retrieve information about the Global and Local Schema and the mapping table
3. once the queries have been rewritten, *Rewriting Agents* send it to the *Mapper Agents*
4. *Mapper Agents* decompose the global queries into a set of local queries
5. once set of local queries is available, *Mapper Agents* send it to the *Planner Agents*

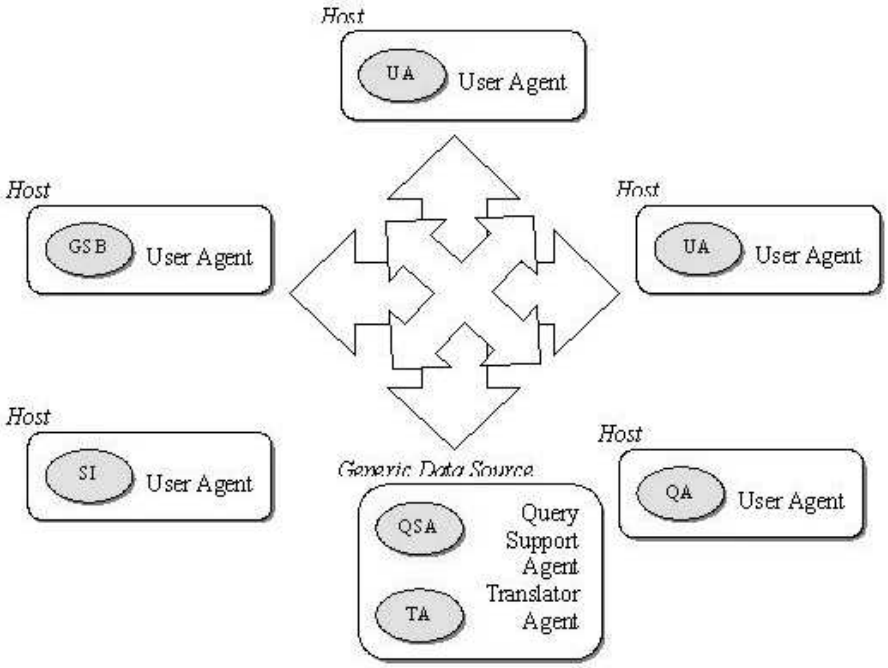

Fig. 6. The *MIKS* agents organised into a fully distributed configuration

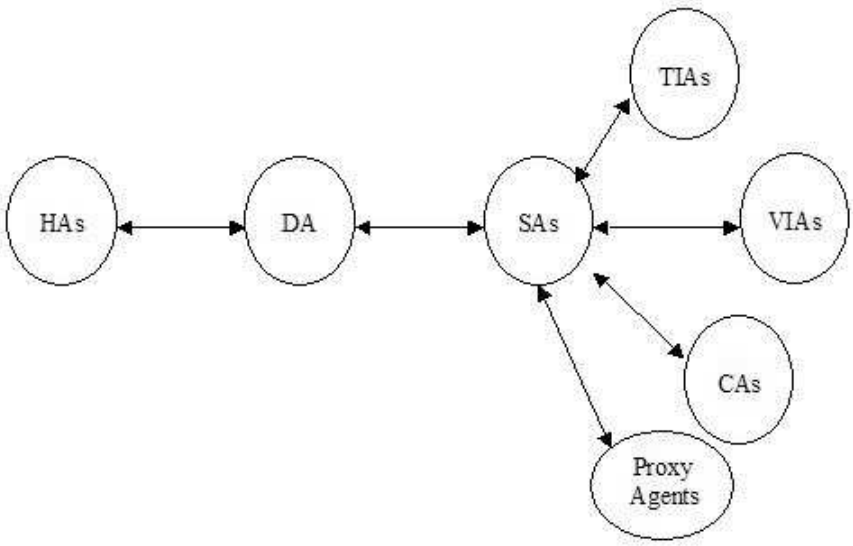

Fig. 7. System initialisation

6. *Planner Agents* generate a query execution plan
7. together with the *SA* organises a group of *Query Agents*
8. *Query Agents* interact with *Query Support Agents* to actually execute the queries at data source level
9. *Query Agents* keep the *Planner Agents* up-to-date, exception should arise and modifications to the plan should be undertaken
10. *Query Agents* synthetise the global answer and send it to the *User Agents*.

These interaction patterns show how our MAS architecture provides for flexibility and can adapt to dynamic environment. Although we have illustrated the *MIKS* system as composed of two MASs, they have to be considered as open system in that agents belonging to either MASs can engage in interactions with all of the other agents. More in general, a system can comprise many MASs of both kinds, building a complex information infrastructure, where many agents interact in order to integrate and query different type of distributed sources and ontologies.

6 Related Work

In the area of heterogeneous information integration, many projects based on a mediator architecture have been developed. The mediator-based TSIMMIS

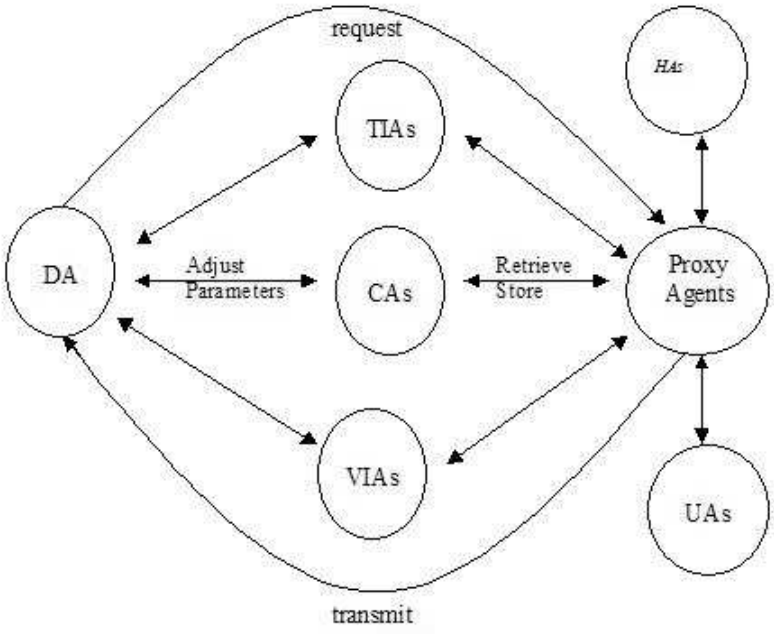

Fig. 8. A possible interaction pattern for the integration process

project [26] 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 our integration approach 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 [21] 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 [3] 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 [47] 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 de-

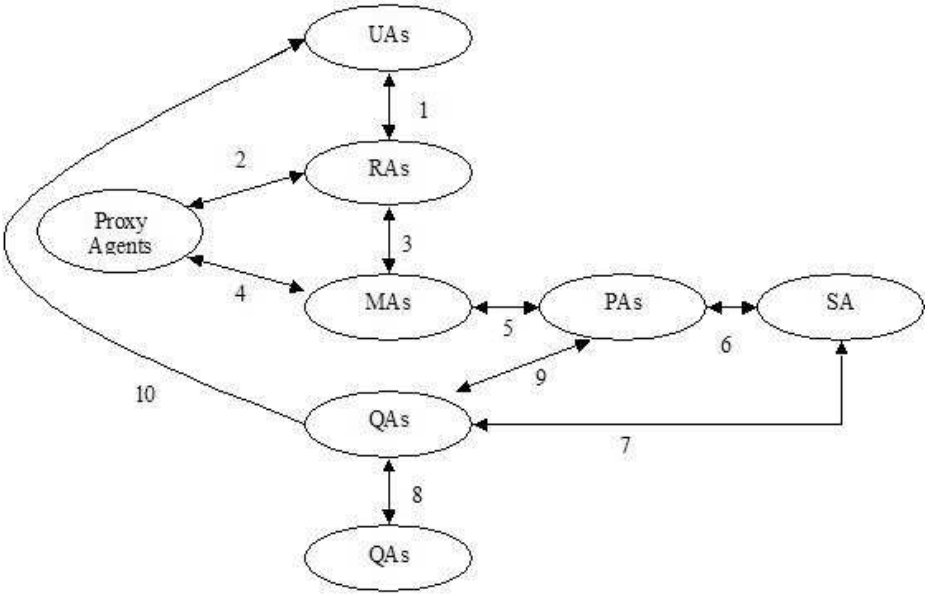

Fig. 9. A possible interaction pattern for the query solving phase

cide 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 [34] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modelled 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. our approach is the lack of a tool aid-support for the designer in the integration process.

As for the multi-agent system community, some work has been done in the direction of integration systems. For its similarities with the *MIKS* system, a particular mention deserves the Infosleuth system. Infosleuth is a system designed to actively gather information by performing diverse information management activities. In [52] the Infosleuth's agent-based architecture has been presented. 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.

While addressing to the same research area, the *MIKS* system and Infosleuth system present slightly different features.

First of all, the scope of the two systems appears to be different. *MIKS* aims at building ontologies related with the integration domain, and at providing a unified view. Query are to be posed as global ones on the GVV. Infosleuth bases its data analysis on given ontologies (rather than building them) and provides visibility of data related only to the specified queries.

Secondly, *MIKS* is meant to provide a two step managing of data, i.e integration and if required also querying, while Infosleuth is devoted to directly query an information source, once an ontology has been explicitly given by humans.

In fact, the integration process differs in that *MIKS* aims at building ontologies directly from the content of the data source, inferring the relationships within the collection of concepts. Infosleuth seems to be more an ontology-driven query engine. Ontologies can be directly provided by users or designers in order to be used during the mapping process. Ontologies can be stored in some server facility for further reuse.

Thirdly, *MIKS* is characterised by strong reasoning capabilities that are meant to tackle the problem of semantic integration of concepts belonging to multiple ontologies (i.e. how we can discover that two objects belonging to different schema refer to the same real-world concept).

Further, as a consequence of these differences, the agent architecture of the two systems is quite different. Agents with common functionalities (translator agents/query support agents and resource agents, user agents, query agents) are still observable even though they reflect the two distinct approaches. One last remark concerns the presence in Infosleuth of service agents (in particular broker agents). The *MIKS* provide the same services in a centralised manner with the *SA*.

Another experience is the RETSINA multi-agent infrastructure for in-context information retrieval [60]. In particular the LARKS description language [61] 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.

References

1. 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.
2. 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.
3. Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.
4. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
5. I. Benetti, D. Beneventano, S. Bergamaschi, A. Corni, F. Guerra, and G. Malvezzi. Si-designer: a tool for intelligent integration of information. *Int. Conference on System Sciences (HICSS2001)*, 2001.

6. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In *VLDB 2000, Proc. of 26th International Conference on Very Large Data Bases, 2000, Egypt*, 2000.
7. D. Beneventano, S. Bergamaschi, A. Garuti, C. Sartori, and M. Vincini. ODB-QOptimizer: un Ottimizzatore Semantico di interrogazioni per OODB. In *Terzo Convegno Nazionale su Sistemi Evoluti per Basi di Dati - SEBD95, Salerno*, 1996.
8. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. The momis approach to information integration. In *Conference on Enterprise Information Systems (ICEIS01), Setbal, Portugal*, 2001.
9. D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.
10. D. Beneventano, S. Bergamaschi, and F. Mandreoli. Extensional Knowledge for semantic query optimization in a mediator based system. In *Proc. of FMII*, 2001.
11. D. Beneventano, S. Bergamaschi, and C. Sartori. Semantic query optimization by subsumption in OODB. In *Flexible Query Answering Systems*, volume 62 of *Datalogiske Skrifter - ISSN 0109-9799*, Roskilde, Denmark, 1996.
12. 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.
13. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. Odb qoptimizer: a tool for semantic query optimization in oodb. In *Fifth Conference of the Italian Association for Artificial Intelligence (AI*IA97), Rome 1997, LNAI 1321*, 1997.
14. S. Bergamaschi, G. Cabri, F. Guerra, L. Leonardi, M. Vincini, and F. Zambonelli. Supporting information integration with autonomous agents. In *CIA*, pages 88–99, 2001.
15. S. Bergamaschi, G. Cabri, F. Guerra, L. Leonardi, M. Vincini, and F. Zambonelli. Exploiting agents to support information integration. *International Journal on Cooperative Information Systems*, 11(3), 2002.
16. 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.
17. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.
18. S. Bergamaschi and B. Nebel. Acquisition and validation of complex object database schemata supporting multiple inheritance. *Journal of Applied Intelligence*, 4:185–203, 1994.
19. J. (eds.) Bradshaw. *Handbook of Agent Technology.* AAAI/MIT Press, 2000.
20. G. Cabri, L. Leonardi, and F. Zambonelli. Agents for information retrieval: Issues of mobility and coordination. *Journal of Systems Architecture*, 46(15):1419–1433, 2000.
21. 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.
22. S. Castano and V. De Antonellis. Deriving global conceptual views from multiple information sources. In *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, 1997.

23. S. Castano, V. De Antonellis, and S. De Capitani Di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2001.
24. R. G. G. Cattell, editor. *The Object Database Standard: ODMG93*. Morgan Kaufmann Publishers, San Mateo, CA, 1994.
25. C.-C. K. Chang and H. Garcia-Molina. Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources. In *Proc. of ACM SIGMOD*, pages 335–346, 1999.
26. S. Chawathe, H. Garcia-Molina, 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.
27. S. S. Chawathe, H. Garcia-Molina, and J. Widom. A toolkit for constraint management in heterogeneous information systems. In *ICDE*, pages 56–65, 1996.
28. World Wide Web Consortium. Standards for the world wide web. Technical report, www.w3c.org.
29. C. Sartori D. Beneventano, S. Bergamaschi and M. Vincini. Odbtools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. In *Conf. on Data Engineering, ICDE'97, Birmingham, UK*, 1997.
30. O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems*, 1997.
31. Oliver M. Duschka. Query optimization using local completeness. In *AAAI/IAAI*, pages 249–255, 1997.
32. S. Gnanaprakasam E. Lambrecht, S. Kambhampati. Optimizing recursive information-gathering plans. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 99*, 1999.
33. FIPA. Specifications, 97.
34. M. R. Genesereth, A. M. Keller, and O. Duschka. Infomaster: An information integration system. In *Proc. of ACM SIGMOD*, 1997.
35. Object Management Group. *CORBA: Architecture and Specification*. August 1995.
36. A. Gupta and J. Widom. Local verification of global integrity constraints in distributed databases. In *Proc. of ACM SIGMOD*, pages 49–58, 1993.
37. L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB*, pages 276–285, 1997.
38. R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.
39. N. R. Jennings and M. J. Wooldridge. Applications of intelligent agents. In Nicholas R. Jennings and Michael J. Wooldridge, editors, *Agent Technology: Foundations, Applications, and Markets*, pages 3–28. Springer-Verlag: Heidelberg, Germany, 1998.
40. C.A. Knoblock J.L. Ambite. Flexible and scalable query planning in distributed and heterogeneous environments. In *Proc. of the 4th Int. Conf. on Artificial Intelligence Planning Systems. AAAI*, 1998.
41. N. M. Karnik and A. R. Tripathi. Design issues in mobile-agent programming systems. *IEEE Concurrency*, 6(3):52–61, 1998.
42. W. Kim and F.H. Lochoisky. *Object-Oriented concepts, Databases, and Applications*. ACM Press. Addison-Wesley Publishing Co., New York, NY, USA, 1989.
43. M. Klusch. Information agent technology for the internet: A survey. *Data and Knowledge Engineering*, 36(3):337–372, 2001.

44. C.A. Knoblock. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95*, 1995.
45. Telecom Lab. Jade - java agent development environment.
46. U. Leser. *Query planning in Mediator Based Informaiton Systems.* PhD thesis, TU Berlin, September 2000.
47. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*, pages 251–262, 1996.
48. S. E. Madnick. From vldb to vmldb (very many large data bases): Dealing with large-scale semantic heterogenity. In *VLDB*, pages 11–16, 1995.
49. Sun Microsystem. Java applet. http://java.sun.com/applets/.
50. Sun Microsystem. Java official web-site. Technical report, www.java.sun.com.
51. Sun Microsystem. Java servlet. http://java.sun.com/servlets/.
52. M. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, and A. Unruh. Active informaiton gathering in infosleuth. *International Journal of Cooperative Information Systems*, 9(1-2):3–27, 2000.
53. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.
54. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.
55. S. Papastavrou, G. Samaras, and E. Pitoura. Mobile agents for WWW distributed database access. In *ICDE*, pages 228–237, 1999.
56. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 484–495. Morgan Kaufmann, 2000.
57. M. T. Roth and P. M. Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In *Proc. of VLDB*, pages 266–275, 1997.
58. M. Ozsoyoglu S. Shenoy. Design and implementation of a semantic query optimizer. *IEEE trans. Data and Knowledge Engineering*, 1(3):344–361, 1989.
59. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
60. K. Sycara. In-context information management truough adaptative collaboration of intelligent agents. *Intelligent Information Agents*, pages 78–99, 1999.
61. K. P. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record*, 28(1):47–53, 1999.
62. M. Wooldridge. *An Introduction to Multiagent Systems.* Wiley, 2002.
63. R. Yerneni, Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Fusion queries over internet databases. In *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology*, volume 1377 of *Lecture Notes in Computer Science*, 1998.