Exploiting Agents to Support Information Integration

G. Cabri[1], F. Guerra[1], M. Vincini[1], S. Bergamaschi[1 2], L. Leonardi[1], and F. Zambonelli[3]

e-mail: {cabri.giacomo,guerra.francesco,vincini.maurizio,
bergamaschi.sonia,leonardi.letizia,zambonelli.franco}@unimo.it

[1] Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905, 41100 Modena

[2] CSITE-CNR Bologna
V.le Risorgimento 2, 40136 Bologna

[3] Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia
Via Allegri 13, 42100 Reggio Emilia

Abstract. Information overloading introduced by the large amount of data that is spread over the Internet must be faced in an appropriate way. The dynamism and the uncertainty of the Internet, along with the heterogeneity of the sources of information are the two main challenges for the today's technologies related to information management. In the area of information integration, this paper proposes an approach based on mobile software agents integrated in the MOMIS (Mediator envirOnment for Multiple Information Sources) infrastructure, which enables semi-automatic information integration to deal with the integration and query of multiple, heterogeneous information sources (relational, object, XML and semi-structured sources). The exploitation of mobile agents in MOMIS can significantly increase the flexibility of the system. In fact, their characteristics of autonomy and adaptability well suit distributed and open environments, such as the Internet. The aim of this paper is to show the advantages of the introduction in the MOMIS infrastructure of intelligent and mobile software agents for the autonomous management and coordination of integration and query processing over heterogeneous data sources.

1 Introduction

Information in the Internet world is spread and highly heterogeneous, and providing an integrated access to multiple sources is a challenging issue concerning cooperation and interoperability in global information systems. 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 of the sources, 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, to structural conflicts. With respect to conventional view integration techniques [4], in the large scale there are complicating factors due to the semantic heterogeneity of the sources. Therefore, 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 the MOMIS (Mediator envirOnment for Multiple Information Sources) infrastructure, which focuses on capturing and reasoning about semantic aspects of schema descriptions of heterogeneous information sources, enhanced by the integration of *mobile agents* for supporting integration and query optimization.

MOMIS [7, 11, 8, 10] 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 [21] 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 a wide and open environment, such as the Internet, MOMIS suffers from some limitations related to the uncertainty and the unreliability intrinsic to large-scale environments. Moreover, a better management of the connections and the exchanges of information help MOMIS in saving network resources. The exploitation of mobile agents can overcome such limitations and increase flexibility. In fact, they well suit the requirements of Internet applications thanks to their characteristics. The agency feature [31] 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 [32] 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 [15].

In this paper we show the advantages 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. In particular:

– agents can be exploited to manage the sources in an autonomous and intelligent way;

- agents, by moving on the source sites, can save bandwidth and deal with unstable connections;
- depending on the queries and on the source site locations, agents allow to choose the best one among different query strategies;
- mobile agents well fit a scenario where users, sources or both can be mobile.

The outline of this paper is the following: Section 2 introduces the architecture of the system and explains the roles of the agents. Section 3 shows how the sources are managed and integrated, by means of an example in the area of car catalogues. Section 4 presents the related work. Section 5 concludes the paper and sketches some future work.

2 The System Architecture

MOMIS provides 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. Like other integration projects [1, 38], MOMIS follows a "semantic approach" to information integration based on the conceptual schema, or metadata, of the information sources, and on the I^3 architecture [30]. In the following we present the extension of MOMIS by integrating software agents (both fixed and mobile) to which delegate specific tasks (see Figure 1). In the rest of the paper we refer to MOMIS as the new infrastructure including agents. The resulting architecture is composed by the following components:

1. a *common data model*, ODM$_{I^3}$, and the corresponding 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 [14]. 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 *Global Schema Builder* (GSB) and the *Query Manager* (QM). The Global Schema Builder 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.

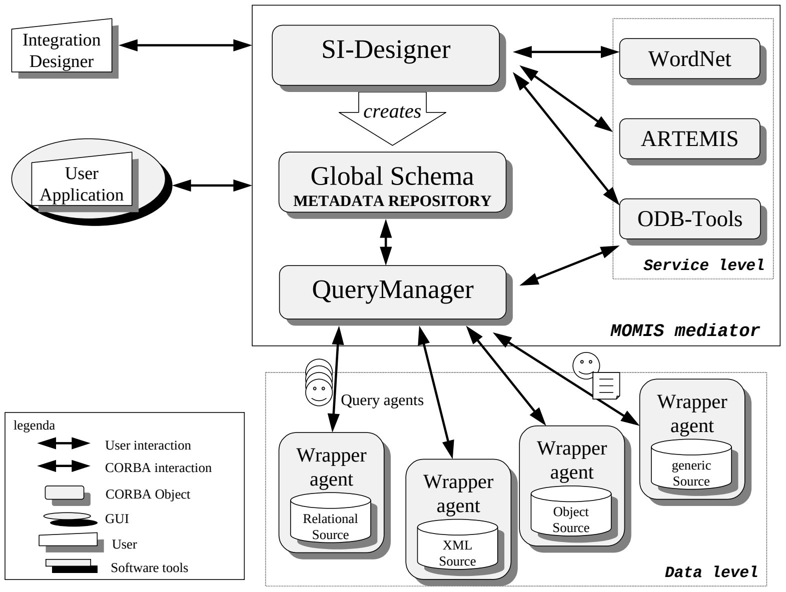

Fig. 1. The MOMIS system architecture

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 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 built by using lexical knowledge from WordNet [29, 34].

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

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

2.1 The Roles of the Agents

The agents exploited by our system are developed in Jade [5], a Java-based agent platform which enables also mobility.

In our architecture, agents have two main roles. On the one hand, the wrappers are agents that convert 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 (represented by rounded boxes in Figure 1). Besides interacting with query agents as described later, the wrapper agents monitor the changes that may occur in the data structure of sources; in fact, the integration of a source is performed once at the beginning, while queries occur later, even after a long period, during which the source may be modified. 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. In this case, the autonomy feature of agents helps in deciding how to deal with the occurred changes. This feature make our system extremely flexible. In fact, an agent-based architecture can be exploited to add flexibility if the mediator system has to manage semistructured sources. As stated previously, semistructured data represent irregular, unknown or often changing structure set of data. An architecture that is able to perceive changes in source schemas and that gives mechanisms to autonomously evaluate the total impact of a change over the whole schema, needs less interactions with the designer.

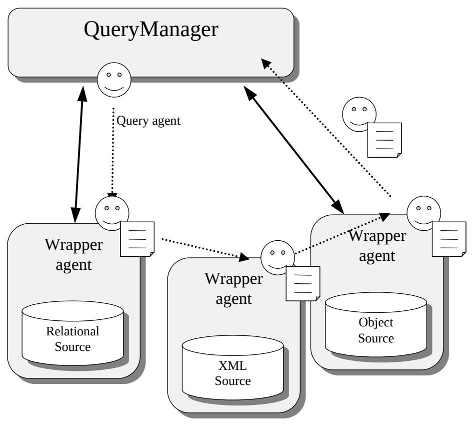

Fig. 2. A trip query

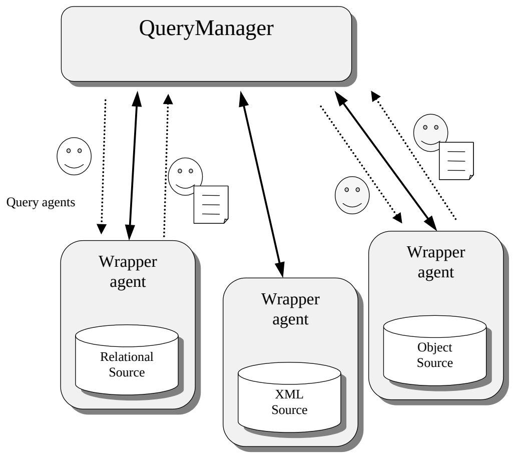

Fig. 3. A parallel query

The QM works as follows. When a user queries the global schema, it generates a set of sub-queries to be sent 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 (see the “moving smiles” in Figure 1). The QM chooses the number of query agents to use by a component that analyzes each query [3]. 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 continues to visit source sites, performing queries on the basis of the already-found information (see Figure 2). 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 (see Figure 3). 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 query 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.

As stated above, agents must interact and coordinate with each other to accomplish their tasks. In particular, an interaction that worth being outlined is the one between the query agents and the wrapper agents.

When a query agent arrives at a site where a source resides, it has to submit its query to the wrapper agent, and, then, to retrieve the results. This interaction can occur by using different protocols. MOMIS agents exploit FIPA speech acts to exchange information, and we think it is the simplest way because it enables to send the wrapper agent a message containing the query itself. However, it is not obvious that such communication mechanism is allowed on every site, because of security and localization issues. Another possible protocol that we are evaluating is based on a shared data spaces where agents put and retrieve messages, uncoupling in this way the interactions and achieving more flexibility. See [17] for a comparison among different kinds of coordination for Internet applications based on mobile agents; an approach that is gaining ground more and more in the Internet-agent area is the one based on *programmable tuple spaces* [16], which enable context-dependent coordination [19].

2.2 Wrapping semistructured sources

During the information integration process, many problems arise due to structural and implementation heterogeneity (including for example differences in hardware platforms, DBMS, data languages) and from the semantic heterogeneity, when different names are employed to represent the same information (naming conflicts) or when different modelling constructs are used in different sources to represent the same kind of information. In the following we explain the approach adopted by a wrapper agent to manage interaction with the sources.

To manage the implementation heterogeneity, a mediator system typically encapsulates each source by a wrapper, which logically converts the underlying data structure to the common data model. Thus, the wrapper architecture and interfaces are crucial, for managing the diversity of data sources [38].

In particular, the main tasks of a wrapper component are:

– during the integration process, the wrapper translates the schema of a source into the common data model of the mediator (in our approach by using the language ODL$_{I^3}$);
– during the querying process, the wrapper converts queries posed over the common data model into requests suitable for the sources, and it converts data returned by the source into the common data model.

For a conventional structured information source (e.g. relational databases or object oriented databases), the schema description is always available and can be directly translated into the selected common data model by the available wrappers. For example, for object oriented databases and flat files, MOMIS wrappers perform a syntactic translation, while for relational databases the translation is based on transformation rule-sets, as described in [24], to perform relational to ODMG schema conversion. For semistructured information sources (e.g., Web data sources), a schema description is in general not directly available at the sources; in fact, a basic characteristic of semistructured data is that they are “self-describing”, hence the information associated with the schema is specified within data [12].

Thus, to integrate a semistructured source, a specific wrapper has to implement a (semi)-automatic methodology to extract and explicitly represent the schema of the source. In [35, 13] methodologies to extract structures from semistructured data files are proposed. Moreover, in order to integrate semistructured information sources, we have to take into account that several data models representing this kind of source have been proposed in literature. In particular, the OEM model (Object Exchange Model) may be thought as the de facto model to represent semistructured data. It was originally introduced for the Tsimmis data integration project [22].

Following the OEM model, MOMIS represents semistructured information sources as rooted, labeled graphs with the semistructured data (e.g., an image or free-form text) as nodes and labels on edges. A *semistructured object* (object, for short) can be viewed as a triple of the form <`id, label, value`>, where `id` is the object identifier, `label` is a string describing what the object represents, and `value` is the value, that can be atomic or complex. An *atomic value* can be integer, real, string, image, while the complex value is a set of semistructured objects, that is, a set of pairs (id,label). A *complex object* can be thought as the parent of all the objects that form its value (children objects). A given object can have one or more parents. We denote the fact that an object `so′` is a child object of another object `so` by `so` $\rightarrow$ `so′` and use notation `label(so)` to denote the label of `so`. In semistructured data models, labels are descriptive as much as possible. Generally, the same label is assigned to all objects describing the same concept in a given source. To represent the schema of a semistructured source S, we introduce the notion of *object pattern*. All objects `so` of S are partitioned into disjoint sets, denoted set, such that all objects belonging to the same set have the same label. An object pattern is then extracted from each set to represent all the objects in the set. Then, an object pattern is representative of all different objects that describe the same concept in a given semistructured source. An object pattern description follows *open world semantics* typical of the Description Logics approach [44]. Objects conforming to a pattern share a common minimal structure represented by non-optional properties, but can have further additional (i.e., optional) properties. In this way, objects in a semistructured data source can evolve and add new properties, but they will be retrieved as valid instances of the corresponding object pattern when processing a query. In the following, we describe an example of wrapper, in particular, in charge of dealing with XML sources.

A wrapper for XML-based files According to the adopted data model (ODM$_{I^3}$), we developed a wrapper to manage XML files. This wrapper aims to map the data model of an XML file into the corresponding object pattern model. This wrapper could be thought as the core of further extensions that aim at managing other XML based files, such as XHTML files.

The Extensible Markup Language (XML) [43] is a W3C recommendation and it arises as a language to describe information sources by using a universal format. One of the main goals of this standard is to exchange files across the Internet. In comparison with HTML, XML can be synthesized as follows [40]:

- The user may define personal tag names at will;
- The structure of the XML file may include tags nested to any level;
- A XML file may include a description of its structure for use by applications that need to perform structural validation. This definition is called DTD (Document Type Definition).

In this way, a XML file may be thought as self-describing like a semistructured data source. The main analogies with our model of a semistructured data source may be summarized as follows:

- object pattern attribute $\longrightarrow$ XML tag
- object pattern $\longrightarrow$ DTD element
- atomic value of an object pattern attribute $\longrightarrow$ PCDATA value

By using this mapping, our wrapper parses the DTD associated to each well-formed XML file and generates a translation from an XML statement into an ODL$_{I^3}$ statement. This mapping implies some critical aspects due to the lack of semantics of XML w.r.t. ODL$_{I^3}$. In particular, the most relevant are: the order in which attributes are described in the DTD, the translation of the concept of attribute from XML language into ODL$_{I^3}$ language, the poor type system provided by XML and the weak semantics of intra-schema references. In the latter case, to avoid loss of information during the translation process, the designer may be asked to supply further information by a graphical interface. XML Schema, a recent W3C standard, allows to express more semantics on structures and datatypes, by using a XML-based language. Our wrapper is able to integrate XML-Schema sources generating a more significant translation in ODL$_{I^3}$, furthermore the wrapper can be extended to manage other XML-based languages (i.e. RDF, XHTML).

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 (*synonymy*, *hypernymy* and *relationship*) between classes and/or attribute names. In this phase, to extract lexicon derived relationships the WordNet database is used.

```
<!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,height,
      luggage_capacity)>
<!ELEMENT performance (urban_consumption,
      combined_consumption,speed)>
<!ELEMENT name (#pcdata)> ...
```

Fig. 4. Fiat database (`fiat`)

2. *Affinity analysis of classes.*
   Relationships in the *Common Thesaurus* are used to evaluate the level of *affinity* between classes, both 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.

3.1 Running Example

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 information about cars of the Italian car factory (see Figure 4).

```
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. 5. Volkswagen database (`vw`)

The second data-source is the `Volkswagen` database (`vw`) reported in Figure 5, a relational database containing information about this kind of car. Both database schemata are built by analyzing the web site of this factory.

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.

3.2 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 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 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 helps 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. Referring to the *Common Thesaurus* resulting from our example, we show some significant relationships (for each relationship, 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*: in this step MOMIS performs reasoning about the Common Thesaurus relationships by exploiting the subsumption and inheritance computation performed by ODB-Tools [9]. In the examined domain ODB-Tools system infers the following relationships:
   ⟨`vw.Model` RT `fiat.dimensions`⟩
   ⟨`vw.Model` NT `fiat.engine`⟩
   ⟨`vw.motor` NT `fiat.car`⟩

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 [10].

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 the two classes 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 systems by the interchange of an XML data file. In addition, the Common Thesaurus is translated into XML file, so that MOMIS may provide a shared ontology that can be used by different semantic ontology languages [25, 23].

In the referring example the following *Global Classes* 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 mapping selection and of the tool SI-Designer which assist the designer in this integration phase see [6].

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 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 *local wrapper agent* in the Vehicle *Global Class* and the *mapping* is performed.

3.3 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 sending 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-query result reported by an agent 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* [36].

For example, the query “retrieve the car name and price for power levels present in every sources”, is expressed as follows:

```
Q: select V1.name, V1.power, V1.price,
   from vehicle V1
   where 1 <= (select count(*)
               from  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 [10], defines the following local queries (`QL1` expressed by SQL and `QL2` by XQuery [39]):

```
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>
```

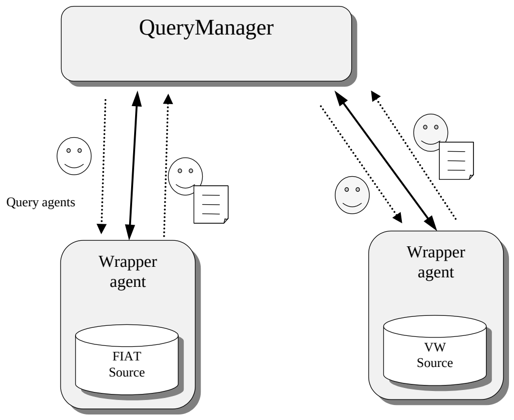

Fig. 6. Parallel queries to two sources with the size of exchanged data

Since the two queries are independent, the QM could assign each one to a query agent, so as the retrieving process is executed in parallel by two agents that move to the two local sources (see Figure 6). After having performed the query, each query agent comes back to the MOMIS site and returns the data result to the QM, which fuses the two data-set obtaining the final result (in the example the fusion is obtained by joining the data-set on the power attribute). The final query is the following:

```
QF: select DISTINCT QL1.name, QL1.KW, QL1.price
    from   QL1, QL2
    where  QL1.KW = QL2.power_kw
    UNION
    select DISTINCT QL2.name, QL2.power_kw, QL2.price
    from   QL1, QL2
    where  QL1.KW = QL2.power_kw
```

Let us suppose that each query (`QL1`, `QL2`) reports an amount of N Kbytes of data results and that the join query `QF` mantains only the 50% of data (N/2). The total amount of transferred data is 2 * N Kbytes, while the ”useful” data is N Kbytes.

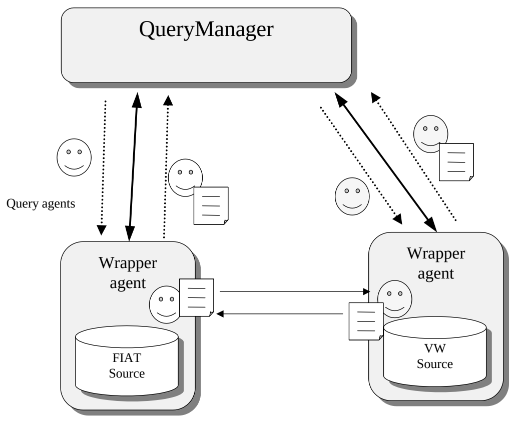

Fig. 7. Query agent interactions can decrease the size of the exchanged data

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 systems where sources reside to minimize the data transfer. In fact, query agents can obtain intermediate results, which are sent to other query agents to perform fusion at the local sites (see Figure 7).

For instance, 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 each query agents to the local sources and the data results (containing only the available engine power) are exchanged with the other agent. Let us suppose that the data amount of these service queries are negligible. In each single source the intermediate data set is 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. Supposing that a short negligible message (the service queries) between the agents implies a transfer of only the ”useful” data to MOMIS, the total amount of transferred data is now about N Kbytes, less of the 2 * N Kbytes of the previous approach.

Moreover, if the sites of the involved sources are “near” (in Internet metrics) one each other but are “distant” from the MOMIS site, one query agent that performs a trip query (see Figure 8 side a) requires only two connection transfers between distant sites (one to go and one to come back), while a traditional client-server approach would require two of such connections *for each remote source* (see Figure 8 side b), thus wasting more bandwidth. For instance, if an American user want to get information about Italian car, all source sites are likely to be located in Europe, while the MOMIS site managing the request is located at the other side of the Atlantic Ocean.

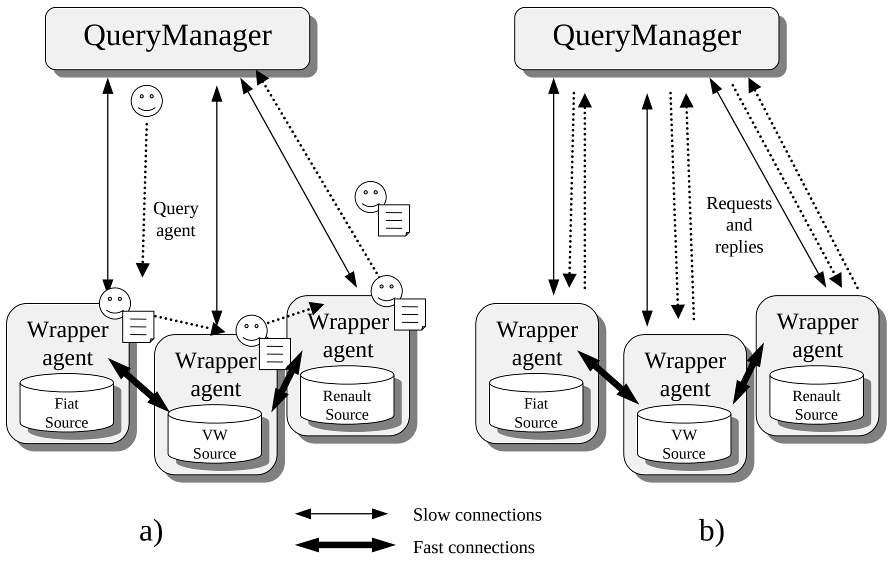

Fig. 8. Agent approach (a) vs. traditional approach (b)

4 Related Work

Although there are several mobile-agent approaches concerning information retrieval [15, 45, 3], demonstrating that mobility can be effectively exploited in distributed information management, there are few agent-based approaches in the area of information integration.

A significative work is the MCC InfoSleuth(tm) [37, 27]. 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 different 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 [41]. In particular, the LARKS description language [42] is defined to realize the agent matchmaking process (at both syntactic and semantic level) by using several different filters: Context, Profile, Similarity, 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 [22] follows a ‘structural’ approach and uses a *self-describing model* (OEM) to represent heterogeneous data sources, *rules* expressed in MSL (Mediator Specification Language) 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.

In the following we present other systems, whose main difference with regard to MOMIS is the lack of a tool aid-support for the designer in the integration process.

The GARLIC project [20] 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 all possible user queries on the schema instead of a predefined subset of them.

The Information Manifold system [33] 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 [28] provides integrated access to multiple distributed heterogenous 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.

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 to integrate 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 modelled 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. Appropriate languages or interaction means should be chosen to grant interoperability and flexibility to agent-based systems. On the one hand, complex languages for the knowledge exchange have been proposed [26]; on the other hand, mobility of agents promotes the adoption of simple and uncoupled coordination protocols [18].

Acknowledgements

Work supported by Italian Ministry for University and Science and Technology Research (MURST) within the project frameworks of “MUSIQUE: Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access” and “D2I: Integration, Warehousing, and Mining of Heterogeneous Data Sources” and AgentLink II - Europe’s Network of Excellence for Agent-based Computing.

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. J. Baek, J. Yeo, G. Kim, and H. Yeom. Cost effective mobile agent planning for distributed information retrieval. In *Proceedings of the 21st International Conference on Distributed Computing Systemsi(ICDCS)*, Phoenix, Arizona, Apr 2001.
4. 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.
5. F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa. Jade programmer’s guide, jade 2.4, sep 2001.
6. 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.
7. I. Benetti, D.Beneventano, S.Bergamaschi, and M. Vincini. An information integration framework for e-commerce. *IEEE Intelligent Systems Magazine*, 2002.

8. 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, Proceedings of 26th International Conference on Very Large Data Bases, September, 2000, Cairo, Egypt*, pages 611–614. Morgan Kaufmann, 2000.
9. 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.
10. 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.
11. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.
12. P. Buneman. Semistructured data. In *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, pages 117–121, Tucson, Arizona, 1997.
13. P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. Adding structure to unstructured data. In *Proc. of ICDT 1997*, pages 336–350, Delphi, Greece, 1997.
14. 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.
15. 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.
16. 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.
17. G. Cabri, L. Leonardi, and F. Zambonelli. Mobile-Agent Coordination Models for Internet Applications. *IEEE Computer*, 33(2):82–89, February 2000.
18. G. Cabri, L. Leonardi, and F. Zambonelli. XML Dataspaces for the Coordination of Internet Agents. *Applied Artificial Intelligence*, 15(1):35–58, January 2001.
19. G. Cabri, L. Leonardi, and F. Zambonelli. Engineering mobile agent applications via context-dependent coordination. *IEEE Transactions on Software Engineering*, 2002. to appear.
20. 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 Univ., 1994.
21. R. G. G. Cattell, editor. *The Object Database Standard: ODMG 3.0*. Morgan Kaufmann Publishers, San Mateo, CA, 2000.
22. 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.
23. DAML Joint Committee. Daml Project. Available at http://www.daml.org.
24. C. Fahrner and G. Vossen. Transforming relational database schemas into object oriented schemas according to odmg-93. In *Proceedings of International Conference on Deductive and Object Databases*, pages 429–446, LNCS Vol 1013, 1995.
25. 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.

26. 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.

27. J. Fowler, B. Perry, M. H. Nodine, and B. Bargmeyer. Agent-based semantic interoperability in infosleuth. *SIGMOD Record*, 28(1):60–67, 1999.

28. M. R. Genesereth, A. M. Keller, and O. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD Conference*, 1997.

29. 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.

30. R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.

31. Nicholas R. Jennings and Michael J. Wooldridge, editors. *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, Berlin, 1998.

32. Neeran M. Karnik and Anand R. Tripathi. Design issues in mobile-agent programming systems. *IEEE Concurrency*, 6(3):52–61, July/September 1998.

33. 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.

34. A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

35. S. Nestorov, S. Abiteboul, and R. Motwani. Inferring structure in semistructured data. *SIGMOD Record*, 26(4):39–43, 1997.

36. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.

37. 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.

38. 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.

39. S.Boag, D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. Xquery 1.0: An xml query language. In *W3C Working Draft 15 February 2001*, Dec 2001.

40. D. Suciu. Semistructured data and xml, 1998.

41. K. Sykara. In-context information management truough adaptative collaboration of intelligent agents. In *Intelligent Information Agents*. M. Klusch Ed. - Springer, 1999.

42. K. Sykara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Records*, 28(1), March 1999.

43. The World Wide Web Consortium (W3C). Extensible markup language (xml). Available at http://www.w3.org/XML/.

44. W. A. Woods and J. G. Schmolze. The KL-ONE family. In F. W. Lehman, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992. Pubblished as a Special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2-9.

45. J. Yang, V. Honavar, L. Miller, and J. Wong. Intelligent mobile agents for information retrieval and knowledge discovery from distributed data and knowledge sources. In *Proceedings of the IEEE Information Technology Conference*, pages 99–102, Syracuse, New York, 1998.