

A WFS-Based Mediation System for GIS Interoperability

Omar Boucelma and Mehdi Essid
Université de Provence, LSIS
39, rue Joliot-Curie
13453 Marseille Cedex 13
France
first.last@cmi.univ-mrs.fr

Zoé Lacroix²
Arizona State University
PO Box 876106
Tempe AZ 85287-6106
USA
zoe.lacroix@asu.edu

ABSTRACT

The proliferation of spatial data on the Internet is beginning to allow a much wider access to data currently available in various Geographic Information Systems (GIS). In order to move to a real Web-based community where geographical data can be accessed and exchanged, we need to provide flexible and powerful GIS data integration solutions. Indeed, GIS are highly heterogeneous: not only they differ by their data representations, but they also offer radically different query languages. A GIS mediation approach should provide (1) an integrated view of the data supplied by all sources, and (2) a geographical query language to access and manipulate integrated data.

In this paper we propose an approach that not only focuses on the data integration, but also addresses the integration of query capabilities available at the sources. A GIS may provide a query capability inexistent at another GIS, whereas two query capabilities may be similar but with a slightly different semantics. We introduce the notion of *derived wrappers* that capture additional query capabilities to either compensate capabilities lacking at a source, or to adjust an existing capability in order to make it homogeneous with other similar capabilities, wrapped at other sources. Finally we describe the implementation of the presented approach that complies with OpenGIS WFS recommendation.

Categories and Subject Descriptors

H.2.8 [Information Systems]: Database Applications

General Terms

Design, Languages

Keywords

Mediation, XML, Geographic Information Systems (GIS), Geography Markup Language (GML), Web Feature Server (WFS).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GIS'02, November 8–9, 2002, McLean, Virginia, USA.
Copyright 2002 ACM 1-58113-591-2/02/0011 ...\$5.00.

1. INTRODUCTION

Geographic data have been collected for centuries and are available in a wide variety of formats and supports. Legacy data remain on paper when other have been digitized. Recent geographic data are available in flat files, databases or Geographic Information Systems (GIS), stored locally or on the Web. Most of the geographic data providers such as the French Institut National de la Statistique et des Etudes Economiques (INSEE)¹, Michelin² or the French Research Institute for Exploitation of the Sea (IFREMER)³ deliver their data in ASCII files (see for example EDIGEO, the road network delivered by Michelin). There is no accepted standard for spatial or geographic data representation. Each GIS provides its own proprietary format as well as its specific query language. In addition, geographic resources are designed for a variety of different purposes. Orthogonal directions in the design of geographic resources may affect the semantics of the data they contain and impair their integration. These discrepancies make the integration of different geographic resources significantly complex.

Recent developments of geomarketing, territory management, and scientific discovery increase the need for integration of these highly heterogeneous resources. With the proliferation of geographic and spatial data on the Internet, a much larger audience is entitled to access and share data currently available in various GIS. Heterogeneity is a key issue for a public or private organization that wishes to consolidate its disparate spatial data into an integrated GIS that would serve as a basis for any decision making process.

The database community has extensively investigated architectures and tools for data integration [18] and developed data warehouses, middleware solutions to facilitate interoperability and data exchange, and heterogeneous distributed DBMS or mediation techniques that facilitate data integration. These approaches usually rely on wrappers [4, 14, 3, 1] to access data sources and to retrieve and translate the results into some common integrated representation. Data warehouses often use wrappers to import data from remote sources that are then materialized locally, and queries are evaluated locally against the warehoused data. A key disadvantage of the warehouse approach is the need for local administrators to maintain the data, while it provides control over the contents of the warehoused data. Mediators and heterogeneous DBMS, on the other hand, submit the

¹<http://www.insee.fr>

²<http://www.michelin.fr>

³<http://www.ifremer.fr>

queries directly to the wrappers, and integrate the results locally. Accessed data is up-to-date, but data access can be costly. In addition, wrappers must be maintained since data providers may change the entry points to the data sources as well as the data organization (schema).

Neither materialized into data warehouses, nor non-materialized approaches address the problem to access or maintain sophisticated tools that enhance data manipulation. Indeed, in addition to standard data manipulation such as performed by SQL, geographic languages usually express spatial selections, metric or topological queries, allocations, etc. [12] and are usually implemented as ad-hoc functions with respect to a specific data representation and indices. These query capabilities may be available partially or totally at some of the data sources. Similar operators may not be semantically equivalent at two different sources. A geographic data warehouse must provide an expressive query language that may be difficult to maintain and may be less expressive than the ones available at the remote sources. On the other hand, non-materialized approaches must not only integrate data, but also all these query capabilities of interest. The approach presented in the paper is a mediation approach that aims to integrate both data and source capabilities.

Most of the existing mediation approaches such as TSMIS [6], HERMES [16], Information Manifold [8], or DISCO [17] focus on the data integration aspects without providing any facility to integrate available tools. While the debate between both approaches (mediation versus materialization) is still open, standard solutions or recommendations are being advocated especially within the Open GIS consortium. Examples of OpenGIS initiatives are the Geography Markup Language or GML [10], and the Web Feature Server (WFS) implementation specifications [11].

Data integration systems are characterized by the way the schemas of the local data sources are related to the global schema. Two approaches [9] exist: Local As View (LAV) and Global as View (GAV). The first approach defines local schemas as views over the global schema. This approach facilitates maintenance. Indeed the addition or removal of a source does not affect the global schema. On the other hand, queries expressed against the global schema need to be reformulated in the terms of the local schemas, that is a costly and complex transformation known to be often NP-complete. To the contrary, GAV offers a straightforward query rewriting at the cost of a difficult maintenance, that may require changes in the global schema.

In this paper we present a mediation system that addresses the integration of GIS data and tools, following a GAV approach. It has a multi-tier client-server architecture based on WFS and uses standard wrappers to access data, extended by *derived wrappers* that capture additional query capabilities. This approach offers several advantages. Derived wrappers can either capture query capabilities available at the source or access a local query capability not available at the source in order to make it available to the user. This approach enables a rich query language in contrast to an approach that would only access the query capabilities available at all sources. In addition, the derived wrapper may adjust semantic differences of similar query operators.

Similar issues were addressed by MOCHA [13] that aims to extend integrated resources with user-defined capabilities. They choose a *query shipping* approach by deploying the needed capabilities as Java code at the remote integrated

sources. Instead, our approach is based on *data shipping*, since the queries submitted to the sources are data collecting queries, whereas the query evaluation is performed locally. In addition, MOCHA is a "full-fledge" middleware system, while our system advocates a mediation infrastructure that complies with emerging standards and OpenGIS recommendation.

This paper is organized as follows. Section 2 describes a motivating example. Section 3 is devoted to the architecture of the GIS mediation system we propose in this paper. Finally we conclude in section 4.

2. MOTIVATING EXAMPLE

Spatial integration is predominantly performed manually, using important human resources [5]. Most current approaches to geographic data integration are data warehouses. This paper does not address the specific issue of schema mapping for data integration. It rather focuses on issues related to the expressive power of query languages.

2.1 Source Schemas

In the sequel, we use an example drawn from [5] which is inspired from the cartographic and topographic databases maintained by the French National Geographic Institute. This example illustrates two road network data sources S1 and S2 describing the same geographic area. Source S1 is equivalent to a 1:250'000 scale, while S2 is more detailed and is equivalent to 1:10'000 scale. Issues raised by the integration scenario are discussed in [5], while more detailed issues on GIS integration may be found in [7].

Figure 1 shows a UML diagram for data sources S1 and S2, together with the integrated schema (IS), as described in [5].

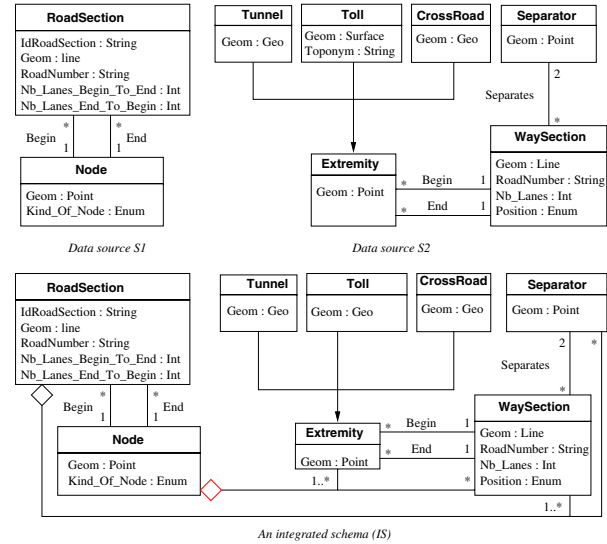


Figure 1: The database schemas

Data source S1 contains the following object types:

- *RoadSection* describes a piece of road, homogeneous in value with respect to the set of attributes and relationships. For instance, changing the number of lanes of a road, results in the creation of a new road section.

- *Node* describes points in the road network where traffic conditions change, hence delimiting road sections.

Data source S2 contains the following object types:

- A *WaySection* is composed by one or several contiguous lane(s) going in the same direction. If the road has several lanes in the same direction split by a separator, then each group of lanes is represented as a separate way section. For instance, a bus lane may be physically separated from two car lanes.
- A *Separator* describes separators between two way sections of the same road.
- An *Extremity* is a point where traffic conditions change. It is either a crossroad, a traffic circle, a toll, a dead end, a value-change node or an end of tunnel. Because the subtypes (crossroad, toll, end of tunnel) have different geometries, the type of an extremity is denoted by geo (any geometry).

2.2 Schema Integration

Consider the XQuery expression (Q) that “extracts all way sections included in the road section “R1”” as defined against the integrated schema in Figure 2. Query (Q) involves two Inter-schema Correspondence Assertions (ICA): C1 and C2, that express inter-schema relationships. More details on these assertions are given in [5]. Clause (C1) specifies that every RoadSection instance corresponds to a multi-sorted set of (WaySection,Separator) instances. Clause (C2) specifies that, for each RoadSection instance, only WaySection instances that satisfy the two conditions below should be considered: (1) match the RoadNumber attribute, and (2) whose geometry lies within a given BUFFER surface enclosing the road section geometry.

```
XQuery (Q):
FOR $W IN document("WaySection")
FOR $R IN document("RoadSection")
WHERE $W/InRoadSection = $R/IdRoadSection
AND $R/IdRoadSection = "R1"
RETURN
<WaySections>$W</WaySections>
```

```
Clause (C1):
RoadSection ⊆ SET([1:N]WaySection, [0:N]Separator)
```

```
Clause (C2):
WaySection.RoadNumber = RoadSection.RoadNumber
AND WaySection.geo INSIDE
BUFFER(RoadSection, resolution)
```

Figure 2: Query (Q) involving ICA rules C1 and C2

The INSIDE operator is the topological relationship which checks whether the point locating the crossroad from S2 is within the area computed by the BUFFER function, which transforms the point representing the S1 node into a surface geometry according to the resolution of data source S2. Note that query (Q) uses (an additional) attribute InRoadSection that results from a UML to semi-structured transformation of the integrated schema. This attribute expresses the fact

that a road section (in S1) corresponds to a set of way sections (in S2).

During the integration process, information is provided either by the data sources, or in computing some operators such as INSIDE: this is a kind of “impedance mismatch” in the sense that there is no data source INSIDE(x,y) to pull data from. Should such a data source be available, the data integration problem would be treated in a uniform way with respect to the mediation solution we discuss in this paper. Our solution relies on a mediation system where resources (operators) are treated as virtual data sources by means of *derived wrappers*.

3. GEOGRAPHIC MEDIATION SYSTEM

In this section we discuss the architecture of our geographic mediation system.

3.1 Mediation Architecture

Figure 3 describes the functional architecture of the geographic mediation system which is mainly composed of three layers: a GIS mediator, Web Feature Servers (WFS) [11] and data sources. The GIS mediator is composed of a query processor which consists of three components: an *Analyzer*, an *Optimizer*, and an *Execution* module. The GIS mediator is in charge of analyzing the (geographic) query, including various transformations that involve ICA rules, performing some optimizations, and splitting the query into sub-queries, passing them to the right WFS for execution.

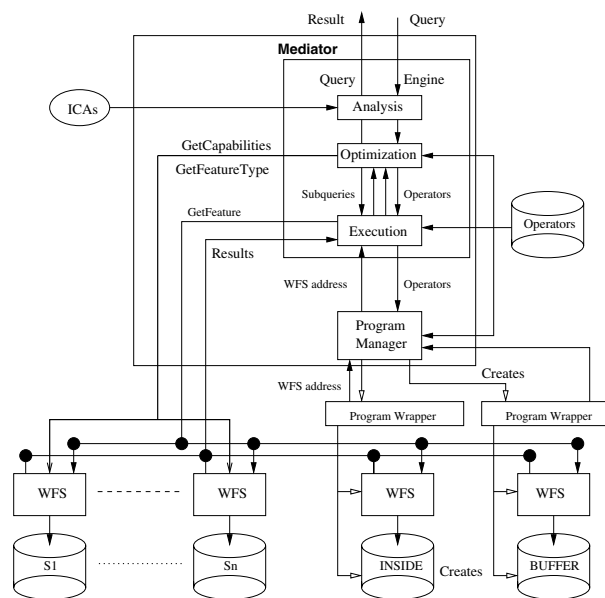


Figure 3: GIS Mediation Architecture

The WFS layer is in charge of manipulating the data sources: it receives requests from the mediator, executes them, and returns the results. A WFS request consists of a description of a query or data transformation operations that are to be applied to one or more features. The request is issued by the client and is posted to a Web server using HTTP. The WFS is invoked to service the request. A request combines operations defined in the WFS specification. Among those operations are:

- **GetCapabilities:** a WFS must be able to describe its capabilities. Specifically, it must indicate which feature types it can service and what operations are supported on each feature type.
- **DescribeFeatureType:** a WFS must be able, upon request, to describe the structure of any feature it can service.
- **GetFeature:** a WFS must be able to service a request, and to retrieve feature instances.

GetFeature requests (posted by the *Analyzer* after transformations) are illustrated in Figure 5. Nowadays, the popularity of XML as a data exchange format, GML designed to represent geographic data, and XQuery [2] makes them good candidates to express respectively schemas, data and queries. The WFS reads and executes the request and returns an answer to the user as a GML or XML document depending on the request.

In Section 1, we have motivated the need for extending standard data integration approaches in order to cope with query capabilities. These query capabilities may need semantic adjustment (when two similar capabilities are not semantically equivalent) or added (when they are not available at a given source). In addition, ICAs may involve operators (such as INSIDE) that are not defined at any data source. Each of these capabilities or integration operators has an implementation locally stored in the *Operators* repository. Its execution is performed by a *Program Wrapper* that creates a virtual additional data source and a WFS which allows the mediator to access to it.

In the sequel, we illustrate the analysis, optimization and execution of query (Q) defined in Figure 1.

3.2 Query Analysis

The *Analyzer* module uses the ICA rules defined in Section 2 to solve aggregation conflicts. The resolution consists in rewriting the query to eliminate the conflicts and to express it against the local source schemas. In our example, we are considering ICA rule composed of two clauses (C1) and (C2). Using (C1), the *Analyzer* discovers the presence of a conflict (the fact that road sections are composed of way sections) in query (Q). This conflict refers to the use of the InRoadSection attribute which is not available at any local source schema, therefore this attribute needs to be replaced with existing local source schema attributes. This task is performed by using clause (C2) that expresses how to solve the conflicts introduced by (C1). The expression containing attribute InRoadSection in query (Q) is then replaced using (C2). This rewriting process results in an equivalent query (Q') as illustrated in Figure 4.

```
FOR $W IN document("WaySection")
FOR $R IN document("RoadSection")
WHERE $W/RoadNumber = $R/RoadNumber
AND INSIDE($W/Geom, BUFFER($R/Geom, resolution))
AND $R/IdRoadSection = "R1"
RETURN
<WaySections>$W</WaySections>
```

Figure 4: Query (Q') after transformations

3.3 Query Optimization

The *Optimizer* component exploits the information about source feature types and WFS capabilities to select an efficient execution plan by dividing the query in several sub-queries. The needed information is provided by WFS in the form of XML and XML schema documents, each time they receive *GetCapabilities* or *GetFeatureType* requests. Using an XQuery processor, the *Optimizer* establishes the execution plan of the query. It splits query (Q) into two kinds of sub-queries: (1) pure XQuery queries that do not contain operators, and (2) operator queries⁴ that cannot be supported⁵ by integrated data sources.

All sub-queries are treated similarly by the WFS layer. A sub-query requiring an operator that is not wrapped at any of the integrated sources is processed by the execution module to create a *derived wrapper* implemented as a WFS as explained in Section 3.4. Since we propose a WFS based architecture, each of the sub-queries is then translated into the query language supported by the WFS, i.e., XML+KVP (keyword-value pair) [11]. An example of a KVP is: "REQUEST=GETCAPABILITIES".

In our example, query (Q) is divided into two sub-queries: Q1 and Q2 to be directly shipped to the WFS that capture data sources S1 and S2, and into parameterized queries: Q3 and Q4 that represent the operators INSIDE and BUFFER locally available. Queries Q3 and Q4 represent XML+KVP document queries that result from a translation of operator queries. XML+KVP queries Q1, Q2, Q3 and Q4 are illustrated in Figure 5.

```
< GetFeature > (Q1)
  < QueryTypeName = "WaySection" / >
< /GetFeature >

< GetFeature > (Q2)
  < QueryTypeName = "RoadSection" / >
  < Filter >
    < PropertyIsEqualTo >
      < PropertyName >
        RoadSection.IdRoadSection
      < /PropertyName >
      < Literal > R1 < /Literal >
    < /PropertyIsEqualTo >
  < /Filter >
< /GetFeature >

< GetFeature > (Q3)
  < QueryTypeName = "BUFFER" / >
< /GetFeature >

< GetFeature > (Q4)
  < QueryTypeName = "INSIDE" / >
< /GetFeature >
```

Figure 5: Query Decomposition

The mediation architecture provides the ability to improve the optimization phase by using additional (metadata) information about sources and operators' capabilities. Iden-

⁴Operator queries are queries containing only specific operators such as INSIDE, BUFFER, etc.

⁵A data source is viewed as a DBMS and its instance.

tifying resource characteristics that can be exploited by the users and the query planning component is critical for the efficiency of the mediation system. We aim to identify metadata along several dimensions, including: (i) the coverage of the information sources, (ii) the capabilities, and (iii) the data delivery patterns of the resources. In the future, useful such metadata could be included in the features supported by WFS.

The step-by-step processing of a query is illustrated in Figure 6. Once the whole rewriting process is done, queries Q1, Q2, Q3, and Q4 are executed as follows.

Sub-queries Q1 and Q2 invoke respectively S1 and S2 to retrieve road sections and way sections. Q1 and Q2 are directly sent to the appropriate WFS by the execution module. When the results of Q1 and Q2 are returned, they are saved respectively in (temporary) QWaySections and QRoadSections GML structures (respectively noted C and T in Figure 6), allowing the system to execute operation BUFFER(QRoadSections,resolution). The *Optimizer* module creates a parameterized XML+KVP document query, then pass the operation to the *Execution* module which executes it and sends to the *Optimizer* needed information to extract the results. The way the *Execution* module executes operator queries and information returned by the *Optimizer* is detailed in the next paragraph. The *Optimizer* finally builds a XML+KVP request Q3 and sends it to a *derived wrapper* implemented by a WFS that will service the request in returning results (noted T' in Figure 6) to the optimizer module. The INSIDE operator query and query Q4 are treated in the same way. Finally, the answer to the original query is built and sent to the user.

3.4 Query Execution

The GIS mediator processes the user's query and forwards the generated sub-queries directly to the *Execution* module as described in Section 3.3. The *Execution* module processes the sub-queries generated by the optimization then it integrates the results returned by the different WFS.

The *Execution* module processes sub-queries as follows. Sub-queries that correspond to existing sources are directly sent to the appropriate WFS. Sub-queries requiring an operator that is not available at any of the integrated sources is processed differently. As each spatial operator is implemented by a *Program Wrapper*, the *Execution* module triggers a job request P1 to execute the operation. Program P1 is processed by the *Program Manager* component which, in turn, builds the associated *Program Wrapper*. A virtual data source is created and populated by results returned by program P1. A *derived wrapper*, implemented as a WFS, is also created in order to access this new data source. Finally, the WFS address is sent to the *Optimizer* module to extract results. It is worth noting that operator sources are accessed through WFS as regular integrated sources.

The architecture described above has several advantages. It first treats equally data generated by applications or contained in data sources. In scientific applications such as geography, it is critical since data is often generated by complex tools as opposed to be hosted in a data source. It is also easy to use the mediation upon sources that are not databases or GIS but can be flat files or Web-hosted sources. Again, in scientific domains, most of the data is actually not stored in database systems. A second advantage of the approach is the use of *derived wrapper* to host

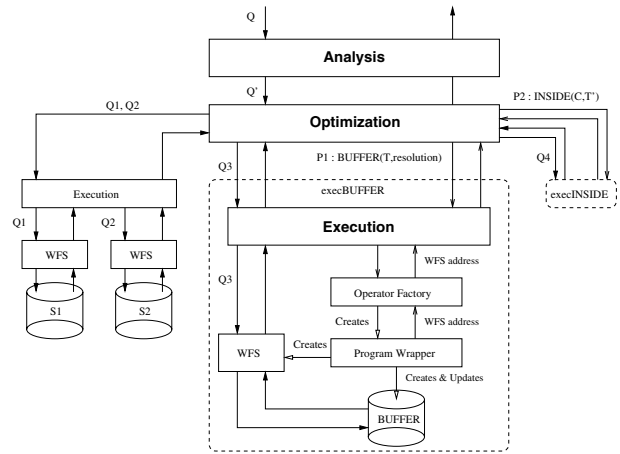


Figure 6: Query Execution

operators that enable the extension of the global mediated system with functionalities that may be missing at some or all integrated sources. With this architecture all data generated by local or remote applications or stored in databases is accessed the same way through WFS. A third advantage is to have a distributed system. Indeed each sub-query submitted by the mediator is considered as a WFS client. Since WFS uses HTTP as a communication protocol, data sources and the mediator may not reside on the same server. Finally, the mediation approach uses commonly accepted format and language, and facilitates maintenance.

3.5 Implementation Details

We are using the GeoServer [15] free implementation of OpenGIS Consortium's WFS implementation specification. As mentioned in Section 3.4, the system uses two kinds of WFS servers: static and dynamic. Static servers handle for available data sources, and dynamic servers handles host (derived) data that are dynamically generated. Each server is implemented by means of a GeoServer instance. To better understand our implementation, let us consider the processing of query Q3 described above. A program wrapper creates a new (derived) data source and its corresponding WFS/GeoServer instance (the server). This is done in three steps:

1. Generating a XML configuration file that contains the parameters that are mandatory for a GeoServer instance: server name, its address, etc.
2. Binding (connecting) the new data source to the server. This is done by creating a dedicated directory schema information and metadata useful for each table in this data source⁶. For example, the following information will be associated to a table <myTable>: the `schema.xml` file is the GML description of the source schema, the `info.xml` file provides <myTable>'s address, a Spatial Reference System (SRS), etc.
3. Instantiating the server. The WFS/GeoServer is now ready to accept queries.

⁶We are using relational tables to host new data sources.

4. CONCLUSION

Geographic data is increasingly becoming available on the Internet, allowing a large audience to access and share the rich databases that are currently used by GIS people. This situation is an opportunity for the emergence of a GIS Web community. However, GIS data is dramatically heterogeneous, being available in various formats, annotated using a variety of methods, and stored in disparate media (flat files, relational or object-oriented databases, etc.). In addition to the scale of data integration, the often complex and heterogeneous query processing and domain specific computational capabilities supported by these sources make GIS integration a real challenge.

In this paper we present a geographic non-materialized data integration system which extends existing data integration systems to accommodate specific GIS operators. The system complies with a standard mediation / wrapper architecture described in the literature, although most of existing systems / prototypes do not deal with spatial data. The implementation we provide complies with OpenGIS standards and specifications such as GML [10] and WFS [11]. We extend previous approaches with the new concept of *derived wrapper*, that is used to cope with GIS operators. A derived wrapper is a wrapper of a virtual source that corresponds to a local data source (buffer) containing results of the execution of a GIS operator. Derived wrappers capture different useful capabilities. First they support data integration by allowing the resolution of different data representation conflicts. In addition, they permit the representation and use of query capabilities that may not be provided by any or some of the integrated GIS systems. As a result, we also tackle the problem of tools integration and extension. Derived wrappers provide a "sound" data integration framework, that is, all the information is captured by means of data sources, wrappers and mediators. Our approach is flexible, since a new source or new source capability may be registered easily with new wrappers (traditional and derived) and ICA rules.

The work described in this paper is part of a larger effort to build an operational spatial data integration system. The future extensions will include: formal and effective spatial transformations, GIS query optimizations, and deployment of the software in a real Web context, leveraging OpenGIS standardization efforts.

Acknowledgment: The work presented in the paper was partially supported by a grant of the French Ministère de l'Industrie via the programme Réseau National de recherche et d'innovation en Technologies Logicielles (RNITL).

5. REFERENCES

- [1] L. Bright, J-R. Gruser, L. Raschid, and M.E. Vidal. A Wrapper Generation Toolkit to Specify and Construct wrappers for Web accessible Data Sources (WebSources). *Journal of Computer Systems Science & Engineering. Special Issue: Semantics on the World Wide Web*, 14(2), March 1999.
- [2] D. Chamberlin, D. Florescu, J. Robie, J.Siméon, and M. Stefanescu. *XQuery: A Query Language for XML*. W3C, 2000. available at <http://www.w3.org/TR/xmlquery>.
- [3] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion. In *Proceedings of the ACM SIGMOD Conference*, pages 177–188, 1998.
- [4] W.F. Cody, L.M. Haas, W. Niblack, M. Arya, M.J. Carey, R. Fagin, D. Lee, D. Petkovic, P.M. Schwarz, J. Thomas, M. Tork Roth, J.H. Williams, and E.L. Wimmers. Querying Multimedia Data from Multiple Repositories by Content: The Garlic Project. In *IFIP 2.6 Third Working Conference on Visual Database Systems (VDB-3)*, Lausanne, Switzerland, March 1995. <http://www.almaden.ibm.com/cs/garlic>.
- [5] T. Devogele, C. Parent, and S. Spaccapietra. On Spatial Database Integration. *International Journal of Geographical Information Science*, 12(4):335–352, 1998.
- [6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaman, Y. Sagir, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and Languages. *Journal of Intelligent Information Systems*, 1997. (See also <http://www-db.stanford.edu/tsimmis/>).
- [7] R. Laurini. Spatial Multidatabase Topological Continuity and Indexing: a Step towards Seamless GIS Data Interoperability. *International Journal of Geographical Information Sciences*, 12(4):373–402, June 1998.
- [8] A. Levy, A. Rajaraman, and J. Ordille. The World Wide Web as a Collection of Views: Query Processing in the Information Manifold. In *VIEWS96 – Workshop on Materialized Views (in cooperation with SIGMOD 1996)*, 1996.
- [9] A. Y. Levy. Logic-Based Techniques in Data Integration. In Jack Minker, editor, *Logic Based Artificial Intelligence*, pages 575–595. Kluwer, 2000.
- [10] OpenGIS. Geography Markup Language (GML) 2.0. see <http://www.opengis.net/gml>.
- [11] OpenGIS. OGC Request 13: OpenGIS Web Feature Server Implementation Specification. see <http://www.opengis.org/info/techno/rfp13info.htm>.
- [12] Ph. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases - with applications to GIS*. Morgan Kaufmann, 2001.
- [13] Manuel Rodriguez-Martinez and Nick Roussopoulos. MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources. *SIGMOD Record*, 29(2):213–224, 2000.
- [14] M.T. Roth and P. Schwarz. Don't Scrap It, Wrap It! A wrapper Architecture for Legacy Data Sources. *Proceedings of the International Conference on Very Large DataBases*, pages 266–275, 1997.
- [15] SourceForge.net. The GeoServer Project. see <http://geoserver.sourceforge.net>.
- [16] V.S. Subrahmanian, S. Adali, A. Brink, R. Emery, J.J. Lu, A. Rajput, T.J. Rogers, R. Ross, and C. Ward. HERMES: A HETerogeneous Reasoning and MEDIator System. TR available at <http://www.cs.umd.edu/projects/hermes/>.
- [17] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to distributed heterogeneous data sources with disco. *IEEE Transactions On Knowledge and Data Engineering*, 10(5):808–823, 1998.
- [18] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, March 1992.