Integrating GIS and Imagery through XML-based Information Mediation

Amarnath Gupta, Richard Marciano, Ilya Zaslavsky, Chaitanya Baru

{gupta,marciano}@sdsc.edu, zaslavsk@rohan.sdsu.edu, baru@sdsc.edu
San Diego Supercomputer Center, 9500 Gilman Drive, La Jolla, CA 92093-0505

1. Introduction

Integration of information from distributed, heterogeneous information sources is an active area of research in the database community. The research efforts have mainly concentrated on sources such as collections of web documents, relational databases and text files. In the domain of spatial information systems, researchers have studied integration architectures, issues and problems in semantic interoperability and information integration concepts for spatial data [0,0, 0]. Some recent approaches [14, 15, 19] have applied information integration methodology from the database community to spatial information systems. In this paper, we propose a mediation-based approach for integrating information from two types of information sources, viz. spatial information systems such as GIS and searchable databases of geo-referenced imagery. As in [14,19], our goal is to enable users to issue a single query in order to search multiple information sources and, in return, receive a combined result incorporating data from across these sources. Similarly, we would like to provide authenticated and authorized users the ability to update sources. This paper describes the architecture of a mediation-based system and steps through the query evaluation procedure in an such a system. We emphasize that the notion of "integration" addressed in this paper does not rest on the development of spatial algorithms that operate on images or vector data and achieve "physical integration" (e.g., see [11]) through techniques like image conflation, as described in, say, [11]. Instead, we aim to attain "logical integration" by creating correspondences between related spatial information similar to non-spatial mediation systems [10]. We demonstrate how existing physical integration techniques can fit into our information association methodology. However, the development of such methods is not the focus of this paper.

1.1 Background

Various interoperability approaches and architectures have been discussed for distributed geographic processing and spatial data integration. Reviews of GIS interoperability and integration efforts are provided in [16, 19, 28, 29]. GIS standardization efforts in a number of countries have resulted in the development of standard specifications for data exchange, including SDTS (U.S.), FEIV (France), ALK (Germany), and SAIF (Canada) [30]. These specifications must contend with *de facto* commercial spatial data interchange standards, such as ESRI's E00 files and shapefiles, MapInfo's MIF/MID, and AutoCAD's dxf. Being relatively new, the former standards have not significantly affected data in legacy spatial information sources. At the same time, these new standards have not linked themselves with emerging standards for Web-based data interchange such as SGML and XML.

GIS integration efforts for supporting spatial data interoperability can be broadly categorized as follows:

- **Cataloguing of geographic sources** (or any sources/datasets), using *locational identifiers*. The Alexandria Digital Library, which supports spatial range queries on a variety of resources, is an example of this approach [ADL];
- **Developing gateways between databases**, by defining universal schemas and persistent views over a variety of data sources .
- **Data warehousing.** Sometimes referred to as the "eager approach" to data integration [14]. The OpenGIS Consortium efforts and related research resulted in the development of GIS interoperability standards based on this model, and in a series of national-level initiatives in the U.S. (via FGDC) and European countries. With several prototypes and testbeds developed (such as the OpenMap testbed [27]), research has focused on semantic and physical interoperability between selected sources. However, experiments of mapping selected GIS data models – Arc/Info, MGE and SPRING - to OpenGIS standard have demonstrated lack of formal standard definition which results in ambiguity and competing alternatives [26]. This approach is efficient for relatively small number of sources with known structure.
- **Mediator-based systems**. Similar to the concept of federated database (also called multidatabases). These systems support homogeneous views (in a common data model) over heterogeneous data sources. Multidatabases are generally based on the client-server model with a middleware system (e.g. based on CORBA or COM) connecting the client and server layers. Mediator systems are based on a 3-level architecture, which include a "foundation" layer (databases with *wrappers*), a mediation layer (which supports exchange of queries and results between wrapped legacy data sources and applications), and an application/user interface layer [10]. The advantage of this architecture is its modularity and scalability. These systems support combining query results from individual sources rather than combining the data. In addition, the use of a semistructured data model at the mediator enables the modeling of sources with no structure or implicit structure. Examples of such semi-structured mediator-based systems include TSIMMIS [1,8], DISCO [23], and Information Manifold [22]. An example of the use of this approach for geospatial data is the Aquarelle project [INRIA] and the research described in [24] and [25]. Accessing geo-referenced SGML-structured information via the Web within the framework of this system is being explored at the time of writing of this paper [31].

Hybrid approaches combining features of the above architectures have also been proposed. For example, a GIS mediation/warehousing architecture described by [13] is built on four layers: the *application* layer (handles end-user requests), the *abstract services* layer (maintains a uniform view of overall system, i.e. a virtual database), the *concrete services* layer (maintains views of precise operations for each system and manages distribution of tasks between systems), and the *system services* layer (invokes services to specialized systems).

1.2 Creating integrated views

An important issue in mediation is the specification of "logical equivalence" relationships among data from different sources. Assume two information sources A and B, as shown in Figure 1. Source A organizes its information elements in the form of a tree while source B organizes its information in the form of a graph.

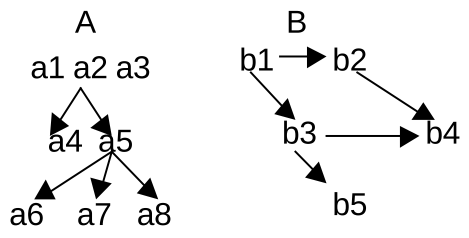


Fig. 1. Information sources A and B are tree- and graph-structured, respectively

The information integration issue relates to defining a *logical* source, say C, from A and B. Source C may never be physically *materialized*, but only generated "on-the-fly". Creating C requires the definition of rules on the original sources such that the elements of C may be defined as a logical association between elements of A and B. In this example, the rules may be:

1. Equate the element *a3* in A with the element *b1* in B
2. Do not include *b5* or any of its descendants in B, in C
3. Make a child of *a5* in A, whose value is greater than 7, also a child of *b4* in B
4. Label elements *a1* to *a8* in A as *c1* to *c8* in C. Also, label elements *b2* to *b4* in B to *c9* to *c11* in C

The resulting "integrated" information source is shown in Figure 2. Note that not all elements of the original sources need appear in the integrated view of the information (e.g. element b5). Information integration is achieved using a set of "association rules".

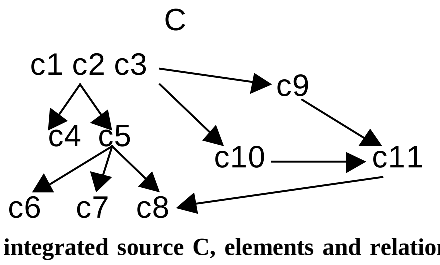


Fig. 2. In the integrated source C, elements and relationships from sources A and B have been integrated according to specific association rules

1.3 Integrating geospatial information sources

Consider two information sources S_1 and S_2 where S_1 is a GIS containing themes such as *soil map, parcel map, digital elevation map* and *transportation network map* of Southern California, and S_2 is an image library containing geo-referenced satellite images, aerial images and property photographs, and associated metadata such as timestamps of images, of different regions in Southern California. Assume that the image library is managed by a DBMS, which is able to provide a complete or a cropped version of any image. The information source S_3 is a view defined over sources S_1 and S_2, using association rules to integrate data across sources. Suppose that S_1 is represented as a tree-structured source using an R-tree, for example, where a node of the R-tree represents the extent of a theme, with additional metadata describing the properties and content of the theme. For the source S_2 , assume that the images are associated with metadata including metadata related to image classification, image segmentation, and annotations. The view exported by source S_2 is a set of trees, where each image corresponds to a tree. Each node of this tree represents a segment of the image, and if node n_1 is a child of node n_2 it implies that the segment represented by n_1 is contained within the segment represented by n_2.

The structure of S_3 is a graph, where the nodes of tree-structured views of S_1 and

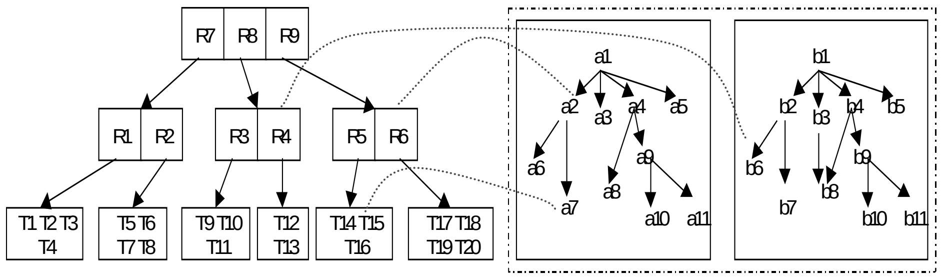


Fig. 3. The structure of the integrated information source. Note the dashed associations between the R-tree-structured representation of the GIS source and the set of image sources.

S_2 are connected through a number of equivalence relations. These relations may be established by rules that specify inter-object associations including, (1) *containment conditions,* e.g., the extent of image object node n_3 in S_2 is covered by theme node n_5 in S_1, (2) *spatial or temporal joins,* (3) *logical associations,* e.g., both items refer to the city of San Diego

2. Architecture of an XML-Based Spatial Mediator

As described before, mediator-based systems employ a 3-level architecture consisting of the application (client) level, the mediator level, and the wrapper level.

Wrappers serve as translators to convert data and query requests between the data model of the underlying information source and the model supported by the mediator. In our approach, we employ an XML-based data model at the mediator level. We extend the mediator of the MIX (***M**ediation of **I**nformation using **X**ML*) project [MIX] to support spatial sources as well. In the MIX system, the result of any query is an XML document. For queries issued on GIS and image sources, the result document may contain text, tables, figures, images, vector graphics and maps. Thus, the mediator should be able to deal with all of these types of information using the XML framework. In general, the MIX mediator receives a user query, fragments the query according to the capabilities of the sources, and sends the fragments to the appropriate sources. As the sources return their individual results to the mediator, the mediator integrates these result fragments into a single combined result and sends that back to the user.

2.1 The MIX framework

Following are the key aspects of the MIX system:

- Each source exports a model of the information it contains in the form of an XML DTD. Data is exported as XML documents which subscribe to the DTD. For both GIS and image sources, the wrapper has to undertake the task of transforming the underlying information into XML. We use XML DTDs as a structural description (in effect, a schema) of the data exchanged by the components of the mediator architecture. The wrappers produce documents that conform to the associated DTD. As described in the next section, the GIS wrapper constructs the DTD by using the "catalog" information in the GIS. The schema provided by a DTD is more versatile than relational schemas, and at the same time provides more structure than the plain semistructured model of existing approaches like TSIMMIS [1,8].
- Each source is queried with an XML-based query language. The XMAS query language [2] has been developed as part of the MIX project. It builds upon ideas from languages such as XML-QL [11], Yat [4], MSL [8], and UnQL [3]. XMAS allows object fusion (e.g., combining an image reference from one source and a map reference from another source into a new composite object) and pattern matching on the input XML data. Additionally, XMAS features powerful grouping and order constructs for generating new integrated XML "objects" from existing ones. The grouping operation can be used to arrange the same information in different ways.
- The query evaluation and integration process may be viewed as generation of a *virtual XML document*. As mentioned before, the output of a query in the MIX framework is an XML document. While it may be possible to materialize this document in one-shot, we provide the flexibility to produce this document in a browsing or navigational mode. In this mode, the user issues an XMAS query, and gets back only a "virtual" unmaterialized result. As the user navigates through the result, the system progressively expands the unvisited parts of the document.

2.2 Extending the MIX framework for spatial data

Figure 4 shows the components of the MIX mediator system specialized for handling spatial information. The XMAS query from the user application is issued to

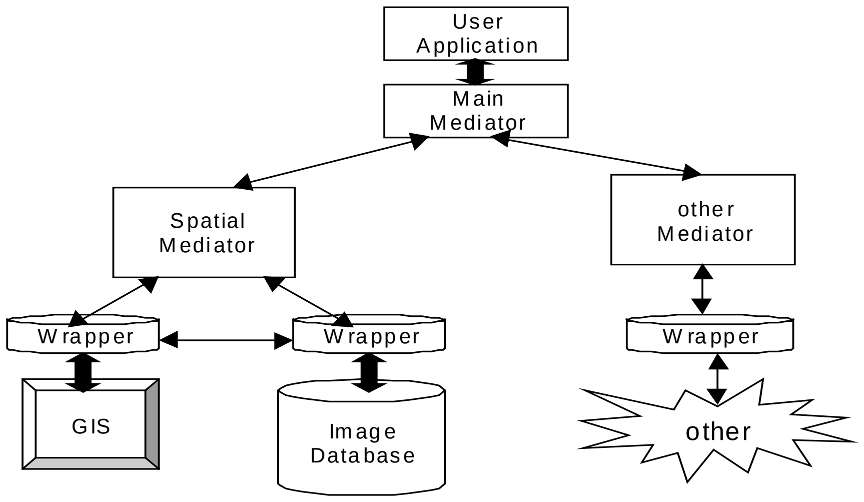


Fig. 4. All the links shown in this figure communicate through XML for data and XMAS for queries. The inter-wrapper links communicate by exchanging binary information if needed

the so-called "main mediator."

The main mediator receives the query containing both the non-spatial and spatial components. It applies a set of rules to identify the spatial part of the query, which it routes to the spatial mediator. In our example, all the query clauses refer to map and image information, thus the entire query is directed to the spatial mediator.

2.3 An example spatial query

As a running example in this paper, we will use the following query based on sources S_1 and S_2 mentioned in the previous section.

Query: Using (a) the *Total Assessed Value (TAV)* (i.e. land price + improvement estimate) of the parcel maps of the regions in San Diego specified as *Carmel Valley* in the 1998 *Police Service Regions* of San Diego, and (b) aerial imagery of the regions and photographs of house properties in those regions, produce the following table:

Table 1. Output of example query

Year	**TAV** > $500K		$300K**<** **TAV** <$500K		$200K< **TAV** <$300K		$100K< **TAV** <$200K		**TAV** <$100K	
1975	Join TAV map of qualified parcels with aerial photo of the same regions	Property pictures of five most expensive properties in the same regions			**Same conditions as in Column 1**					
1980										
1985					Same conditions as for Year 1975					
1990										
1995										

Processing this query requires:

- finding the region corresponding to “Carmel Valley” from the Police Service Regions of 1998
- for each specified year, classifying this region into the five land price ranges as shown, and generating one map per range
- for each map in a given range, overlaying it on top of the appropriate aerial image of Carmel Valley
- for each sub-region, identifying house properties in the region, ranking them by price, and choosing the top five photos
- arranging the information in the requested tabular form, as shown

The query produces a table of 25 maps and 25 sets of house photographs. We chose this particular query in order to show that the query output need not always be a single map. In this case, it is a table of 25 maps. The query demonstrates the need for the mediator to decompose a single query into multiple simpler queries to be executed by the GIS. The query also requires physical integration (the images from the image source need to be fetched and overlaid onto the TAV map) and logical integration, exercised through the join conditions on the aerial images and maps and on the property picture addresses and the qualifying regions in the GIS.

Grouping of the results according to time and value range is achieved using the powerful grouping construct available in the XMAS query language [2]. In the example XMAS query shown in the next subsection, we illustrate how reserved namespaces (i.e., a set of tag names) can be used to tailor XMAS to process data in different domains, e.g. the spatial domain. This query also illustrates the power of the virtual XML document concept. In this example, the query result contains 50 properties in Carmel Valley. The user may be interested in further examining the textual metadata of only, say, the first 20 of these and, further, may retrieve the images of only three properties. The virtual document concept will avoid unnecessary computation of the entire result set.

2.3.1 An Example XMAS Query

The XMAS version of the example query is shown below. Rather than explaining each step of the query, we point to a few key elements:

1. The notation *mix:<tagname>* refers to tags in a reserved namespace, which the mediator is aware of. Thus, the mediator recognizes the type of data and has specific ways of handling that data type. For example, *mix:region* would not be expected to have, say, a *length* attribute.
2. The query is directed to the mediator without specifying the location of any source. The reserved namespace tag *mix:source* (line 10), is used to indicate which definition of "Carmel Valley" should be employed.
3. The function *category(price,totalValueCategory)* (line 43), is a function that examines the total assessed value of a parcel in the parcel map and assigns the parcel to the correct bracket.
4. The output table is defined by grouping the results first by year (using the notation {$y} on line 37) and then by total assessed value ({$c} on line 36) within the year.
5. Wherever the same variable (e.g., $r) is used multiple times, it has to bind to the same constant. Thus, the aerial image, corresponding to *$r* on line 23, and the map object, corresponding to *$r* on line 10, must be of the same region, and the address of the property (line 32) must belong to the parcels satisfying the category condition.
6. The spatial predicate *within(region1,region2)* (line 46) is used without any software dependent syntax. We assume here that the user can enquire from the mediator what the supported functions are and how they can be invoked. For example, if one of sources underneath the mediator may support another function centroid_within(region1, region2), the mediator will export the function and the user has to know which is suited for a query.
7. The predicate *display_order(mapData1, mapData2)* (line 49) is procedural and specifies that mapData1(corresponding to a theme) should be overlaid on mapData2. In case multiple mapData were involved, the mapData elements would be presented as a nested list in the form—display_order(mapData1, (mapData2,mapData3)).

```
1. answer = construct $A
2. where
3. $A:<table>
4. <row>
5.      <year>$y</>
6.         <totalValueCategory>$c
7.           <totalValue>$tv</>
8.           <mapColumn>
9.             <mix:map>
10.              <mix:region mix:source=$s1>$r
11.                <mix:regionName>$n</>
```

```
       </>
       <mix:mapData>$md1
         <mix:dataName>$d1</>
         <mix:dataValue>$tv</>
         <mix:region>$r2</>
         <mix:date>$y</>
       </>
       <mix:mapData>$md2
          <mix:datatype>$dt
          <mix:resolution>$res</>
       </>
          <mix:region>$r</>
          <mix:date>$y</>
      </mix:map>
     </mapColumn>
     <pictureColumn orderby=$p orderType=asc topN=5>
      <mix:image>
       <mix:dataName>$d3</>
       <price>$p</>
       <address>$a</>
       <mix:date>$y</>
      </>
     </>
    </totalValueCategory> {$c}
   </year>{$y}
  </row>
</table>
in http://some.mediator.url
and
belongsTo($y, (1975,1980,1985,1990,1995)) and
category($tv,$c) and  ($s1 =
“San_Diego.Police_Service_Region”) and ($n =
“Carmel Valley”) and ($md1= “Parcel Map”) and
($d1= “total assessed value”) and within($r2,$r)
and ($md2 = “imagery”) and ($dt= “aerial”) and
($res <= 16m) and ($d3 = “property photo”) and
mapsTo($a,$r2) and display_order($md1,$md2)
```

3. Wrapping spatial information sources

In general, queries received by wrappers can be classified as *direct*, *logically equivalent*, or *indirect queries* [1]. A *direct query* is a request that can be satisfied by a primitive operation provided by the underlying information source. *Indirect queries* are those that are not supported by the underlying information source, thus the wrapper itself must have the computational capability to produce the required results. For example, a source may not be able to compute the correlation coefficient of a sequence of number pairs, thus the wrapper would have to perform that computation. Finally, *logically equivalent queries* are those that cannot be directly processed by the source, however, it is possible to rewrite the input query into one or more other queries to the underlying source which, in effect, can answer the original query. For example, the query, “For each census tract in San Diego that has over 30% minority population, find the zip code boundaries that intersect with it,” may require multiple requests to a GIS source to identify the census tracts and then intersect with the zip code regions. However, it is possible to write the GIS script to generate the necessary result, thus, this is an example of a logically equivalent query. The task of the wrapper is further complicated for such queries since it may have to *compose* a program from smaller modules to produce the result. In this paper, we focus primarily on direct and logically equivalent queries, and provide only a simple example of indirect queries.

A key feature that distinguishes GIS sources from many other sources studied in the mediation literature is that most GIS sources are *stand-alone, interactive* systems. At best, interoperability is supported only within the same family of software products and not across federated, heterogeneous GIS sources. A GIS serves the roles of a database system (for spatial data), a computation source (e.g., for network flow optimization), and a presentation source (e.g., surface generation and mosaic creation), and is not usually equipped with a generic query language for declarative access. A GIS wrapper overcomes this problem by maintaining an internal model, including schema as well as instance information of a given source, and simulating the necessary ad hoc interface. The wrapper functions by first transforming the query/result into this internal model.

3.1 The internal schema of the GIS wrapper

Most GIS sources recognize various map layers (e.g. coverage and themes) and a large body of common operations such as overlay and spatial intersections. These ubiquitous objects and functions can be considered as a simple typed algebra (as has been done more than once since 1983 D. Tomlin’s Map Algebra []). For a specific GIS source the wrapper has to know how the types and functions in its algebra maps to the types and functions in the source. The wrapper assumes that there are at least 2 kinds of functions: boolean returning functions and object returning functions. An example of the latter is, say, the function *within*(region1, region2) written in ArcView Avenue script, where region1 is a theme object and region2 is region object, and the function is called #FTAB_RELTYPE_ISCOMPLETELYWITHIN, and needs to be invoked as:

```
  region1.SelectByTheme( region2, #FTAB_RELTYPE_ISCOMPLETELYWITHIN, 0,
#VTAB_SELTYPE_NEW).
```

Using the MapBasic syntax of MapInfo, the same function would be written as:

```
    SELECT * FROM Region1 WHERE OBJ ENTIRELYWITHIN Region2.OBJ
```

The results returned are mapped back to internal types and hence are easily translated into XML. All additional types and operations supported by a GIS system will have to be layered on top of the simple algebra.

The GIS wrapper models every GIS source as having a **collectionObject** at the top level followed by **themeObject** and a **dataObject**, at the next level below. A collectionObject represents a group of co-registered themeObjects. A themeObject has the subtypes:

- **themeMap**: a binary blob that represents a map produced by the underlying GIS as the result of an operation. Each themeMap object has an identifier, a resolution and an extent. It may contain additional metadata, such as the time when the theme was created. A themeObject can be instantiated as a map.
- **table**: a representation of attribute information associated with a GIS theme in the form of a possibly nested table. Each table object has an identifier and a list of column names. A nested table is represented by treating a column name as a named list instead of a singleton name.
- **themeProperties**: a set of properties that describe an aggrgegate of the data contained in the theme. For example, for each type of dataObject in a theme, it will contain information like the number of information items in the theme, the spatial granularity of a cell, indexes implemented if any, existing topology if any, and so forth.

A dataObject represents the type of information associated with any theme. Each data object maintains its spatial extent, and has a reference to the table in the themeObject related to it. It may additionally have information on the valid time of the underlying data and accuracy of the data items. For many common GIS applications, the set of dataObject subtypes includes:

- **regionObject**: representing a 2D polygonal object in a theme
- **curveObject:** representing open or closed polylines in a theme
- **networkObject**: representing a graph possibly consisting of intersecting polylines
- **pointObject**: representing a feature that is represented in a theme as a point object
- **matrixObject**: representing a feature where every element in an array is associated with a value

The purpose of defining object types, like those described above, is to facilitate the translation of an XMAS query into syntactically well-formed queries in the language of the underlying GIS, and to convert the results returned from the GIS into a well-formed XML structure. These object types can be specialized, using inheritance, when developing wrappers for specific GIS systems. For example, the wrapper of a specific GIS, may have chosen to define an object, say, networkObject, as a transportation. For every built-in or user-extended type a list of function signatures is also defined. For example, for curveObject, the spatial intersection is defined as: intersect(curveObject, regionObject) and so on. The function signatures are needed so that the parameters specified therein can be made visible to the mediator[1] and be used to formulate valid XMAS queries. This also enables the wrapper system to be extensible so that if a specific GIS system permits some special function (e.g., water drainage computation for a digital elevation model) or object type, it can be simply exported to the mediator.

[1] This actually happens during a registration procedure, but such operational details are omitted in this paper

3.2.1Catalog Extraction

The catalog is the schema information of the GIS source, which the wrapper exports to the spatial mediator as an XML DTD. It also maintains an internal version of the catalog to translate XMAS queries into GIS queries. In this paper, we use ArcView from ESRI as an illustrative example to discuss catalog services and query translation.

An example XML DTD for the GIS catalog is given below:

```
<!ELEMENT arcview_project (views|tables|scripts)* >
<!ELEMENT views (view)* >
<!ELEMENT view (projection|units|themes)* >
 <!ATTLIST view name CDATA #IMPLIED>
<!ELEMENT projection (#PCDATA)* >
<!ELEMENT units (#PCDATA)* >
<!ELEMENT themes (theme)* >
<!ELEMENT theme (assoc_table|threshold|extents)* >
 <!ATTLIST theme name CDATA #IMPLIED>
<!ELEMENT assoc_table (#PCDATA)* >
<!ELEMENT threshold EMPTY >
 <!ATTLIST threshold val CDATA #IMPLIED>
<!ELEMENT extents (bottom|left|top|right)* >
<!ELEMENT bottom (#PCDATA)* >
<!ELEMENT left (#PCDATA)* >
<!ELEMENT top (#PCDATA)* >
<!ELEMENT right (#PCDATA)* >
<!ELEMENT tables (table)* >
<!ELEMENT table (col)* >
<!ATTLIST table name CDATA #IMPLIED>
<!ELEMENT col EMPTY >
 <!ATTLIST col alias CDATA #IMPLIED>
 <!ATTLIST col type CDATA #IMPLIED>
 <!ATTLIST col width CDATA #IMPLIED>
 <!ATTLIST col decimal CDATA #IMPLIED>
<!ELEMENT scripts (script)* >
<!ELEMENT script EMPTY >
  <!ATTLIST script name CDATA #IMPLIED>
```

In a given GIS instance, we distinguish between a base theme set **B** and view theme set **V** as follows. A theme *b* is a base theme if it has only one of the subtypes of dataObject (e.g., if it only contains regionObjects) or if the wrapper engineer designates it to be a base theme. A theme *v* is a view theme if it has been created based upon a set of base themes $\{b_i\}$, such that the information in *v* is strictly a subset of the information in $\cup_i\{b_i\}$. The intuition behind making this distinction is that it is more optimal if query can be answered from a view theme rather than a base theme, since a view theme is equivalent to materialization, and reuses precomputed expensive spatial predicates on the same base data set. In order to recognize a theme as a view theme we employ the following. Assume that the projects in the system comprise universe of themes[2]. We traverse the project structure of ArcView and identify all themes referenced by it. For themes that have creation scripts, we identify the names of other themes, and arrange them to form a dependency graph of themes which is maintained by the wrapper. All themes with no incoming edges in the graph are placed in **B** and the others are placed in **V**. If a theme does not have a creation script it is placed in **B**. Hence in the worst case, every theme is treated as basic. In addition to this labeling, the wrapper engineer has the ability to specify for each view theme how the view was derived. The derivation needs to state which attributes (spatial or otherwise) were used to derive the view and what restriction condition was applied on the respective attributes. While it is difficult to extract this derivation information automatically, such a specification can enhance the efficiency of query processing. To see why this is so, consider a user query that looks for the ethnic distribution of all census tracts that overlap "Carmel Valley". Let us suppose the wrapper has identified two themes whose tables have the attributes "census tract number" and population. If it is known a priori that one of those two themes has already been restricted (subset) based on another attribute (e.g., median income greater than $25,000), then this information can be used to discard that theme, because the theme will not produce a complete answer (since it does not have the records corresponding to the population having median income less than $25,000). In the absence of such view information, the wrapper has to use other heuristics, such as selecting the theme with a higher record count. We will revisit this issue in a later section.

For each theme object we create a catalog record having the XML DTD structure shown before. Note how the internal schema of the wrapper has been used in constructing and that the themes are labeled as base and view. This basic structure can be augmented by any additional information that can be extracted by traversing the project structure. For example, for ArcView, if a satellite image stored as a layer will have the additional attributes such as "bandstatistics". Instead of simply exporting a set of theme DTDs to the mediator, we organize them into a container document by first creating an R-tree corresponding the spatial extents of the themes and then generating the XML document from the R-tree, as shown in Figure 3. Note that an internal node of the R-tree only induces a nesting in the XML document, without producing material data. The reason for having the R-tree representation at the spatial mediator is to gain efficiency during query processing. It is very likely that in order to choose the candidate sources for a query the mediator will need to ask, "which are the themes that provide some information in the user specified rectangle of interest?" Having the R-tree index within the mediator saves the trouble of going back to the information sources.

In addition to the containment relation and the derivation dependency graph, the wrapper may maintain other indices to connect the themes. One important thread is to place all themes in a temporal order, based on the valid time (i.e., the time when the data items were valid) of the theme. In case of themes valid over an interval of

[2] It is very easy to include themes not referenced by any project in the universe.

time the temporal order may be implemented through a data structure like the interval tree.

3.2 Wrapping an image Library

Image sources may have both metadata-based retrieval and content-based retrieval capabilities [21]. The objective in wrapping image libraries is to provide access via a uniform interface to complete or partial digital images, as well as image features and other associated metadata.

3.2.1 The internal model for image sources

In general, image wrappers recognize the following object types associated with images:

- **image**: an image is associated with a set of standard metadata, e.g. its dimensions, format and pixel depth (bits per pixel). Images may be multiband, and could be retrieved one band at a time or up to three bands together. Standard image operations include cropping, rotation, and changing brightness and contrast. A spatial image is a specialization of image that must additionally have a georeference and resolution.
- **image mask**: a mask is a pixel chain within a bounding box, with specified coordinates. The purpose of a mask is to represent a segment produced by an image processing operation. The purpose of treating an image mask as a distinct data type is to separate the segment and its properties from the image. This allows an image to be associated with multiple segments produced by different operations that can be transferred across different components of the integrated system such as from the image library to a GIS wrapper.
- **image feature**: an image feature is a representation of an image property such as texture in a photograph or concrete region in a satellite image, that is computed by some analysis operation. Each instance of a feature is associated with an image mask that localizes the area over which it was computed. For convenience, a feature instance is associated with additional metadata such as the name of the feature and the parameter values used to compute it. We expect that image libraries will provide similarity functions based on features (e.g., it can request all images having some segment with texture similar to a given texture [20]).
- **scene graph**: scene graph, a term used in the VRML and MPEG-4 literature, represents a tree-like decomposition of a real or virtual scene, using a well-defined system of node types. We do not use all the features of a scene graph like the image transformation specifications. In our usage, a leaf node in the scene graph stands for a “unit region” in the image whose property can be described by a set of simple image features. We also keep the provision that the region defining a leaf node can be described by a shape property (e.g., “a circular area”), where the property belongs to an allowed type in VRML and MPEG-4 scene graphs. An internal node is constructed using the containment relation, as shown in Figure 2.

4. Evaluation of spatial queries

The task of the spatial mediator is to parse the spatial subquery, of a given query, and generate the associated evaluation plan. The spatial mediator is required to, (1) fragment the subquery between information sources and determine the order of execution of each fragment, (2) use the schema and capability information exported by each source to rewrite subquery fragments such that each source is able to evaluate the rewritten fragments, and (3) send each rewritten fragment to the corresponding source, collect the results from each source and return the combined result to the application mediator.

4.1 Query planning at the mediator

In this section, we sketch the typical sequence of steps executed by the spatial mediator. The exact sequence and details of each step will vary depending on the actual query. In the following, we refer back to the query example discussed in Section 2.3.1.

1. **Determine Map Request**: The tag *mix:map*, in line 9 of the XMAS query, specifies that the output of the query should be a map. The spatial mediator then expects to know which geographical area needs to be mapped, and which variables should be used to produce the map.
2. **Identify Map Region**: The next tag *mix:region*, on line 10, specifies the region to be mapped and informs that this region can be found in the source theme called, “Police Service Region” within the provenance of “San Diego”. In this example, we assume that the information mapping the region to the theme is also available in the GIS itself. The mediator searches the DTD exported by the GIS wrapper and determines that the “San Diego City” theme has an associated table called “Police Service Region”, and that the table has a polygon as a field.
3. **Produce Wrapper Query Condition for Map Region**: Line 11 identifies the *regionName* variable, $n, and line 44 specifies that its value must be “Carmel Valley”. The mediator also knows that the tag *regionName* maps to the field name *srvRgn* in the “Police Service Region” table of the source GIS. Thus, it places the query condition *srvRgn= “Carmel Valley”* in the query fragment to be sent to the GIS wrapper.
4. **Identify Map Attribute 1, *total assessed value***: The tag, *mix:mapData* on line 13, specifies the element to be mapped. As before, this data element is identified with a query condition on the “total assessed value” field of the table in the “Parcel Map” theme. But in this case, several “Parcel Map” themes are found, each associated with a different year. The mediator picks the years corresponding to the query by inspecting the DTD for the Parcel Map themes. We will show in the next section how the year gets associated with the theme in the DTD provided by the wrapper. Since 5 years (thus, 5 themes) are requested, the mediator produces 5 (almost identical) copies of all query conditions gathered so far, and treats them as *independent subqueries* to be sent to the GIS wrapper.
5. **Produce Wrapper Query Condition for Map Attribute, *total assessed value***: The *total assessed value* attribute must satisfy the function

category(price,totalValueCategory) and the results must be grouped into columns based on category value. Since there are 5 possible categories that each parcel map can belong to, a map needs to be produced for each. Thus, the mediator issues 5 "related" queries to the wrapper for each independent subquery mentioned above.

6. **Identify Map Attribute 2, *imagery***: The next tag, *mix:mapData* on line 19, specifies the next element to be mapped. This element is an aerial map in the image library and must satisfy query conditions on the year, the resolution and the georeference to identify images for the query. The mediator has the DTD exported by the image wrapper. Whether the query can be safely answered is unspecified at this point since the region specifying the georeference is not computed yet. Thus, the mediator forms a partial query for the image wrapper.
7. **Formulate Query Fragments for Wrapper**: The *mix:map* tag on line 9 has a corresponding end tag on line 26. The mediator produces one independent and several dependent query fragments for the subquery represented by the *mix:map* element.
8. **Determine Image Request**: Similar to the map request, the mediator verifies that the necessary tables and field names exist in the DTD specified by the image library. It uses a rule to determine that the predicate *mapsTo(address, parcel region)* on line 48 is to be performed in two steps: first, obtain the address block from the Parcel Map table in the GIS source, second, formulate a range query on the street number in the image library.
9. **Determine Query Execution Plan**: The query is executed in the following order:
    1. Determine the extent of the Carmel Valley region.
    2. Use the extent to find corresponding aerial images of the region in the image library.
    3. Validate that there exist images of the region at the specified resolution. If there are images at multiple resolutions, use a rule to choose the finest resolution image covering the entire map region.
    4. Initiate transfer of the image from the image library to the GIS[3].
    5. Execute the map retrieval query (which produces 25 sets of results), and determine the parcels. Also, fetch the address block of the parcel, since that will be required in a later part of the query.
    6. Formulate the image query including the sorting.

4.2 Examples of a rewritten query fragment

As mentioned before, a query fragment is a portion of the spatial query that is rewritten to match source capabilities. Each fragment is sent to an individual wrapper. This rewritten fragment contains the table and field names, structures, and functions supported by the source. We illustrate this in the following paragraphs.

- **Determine the boundary of the region Carmel Valley**

```
ans1 = construct $R
where
<ArcView_Projects>
          <theme name = $n1>
```

[3] Assuming for now that the image and the GIS layer can be aligned.

```
          </theme>
          <tables>
            <table name=$n2>
              <col name=$n3>$v1</>
              <col name=$n4>$v2
$R:             <extents>
                  <top>$t</>
                  <bottom>$b</>
                  <left>$l</>
                  <right>$r</>
                </extents>
              </col>
            </table>
          </tables>
</ArcView_Projects>
in http://wrap.gis.url
and ($n1= "sdcity.shp") and ($n2 = "Attributes of
sdcity.shp") and ($n3 = "Police Service Region") and
($v1 = "Carmel Valley") and ($n4= "Shape") and
getExtentTop($v2,$t) and getExtentBottom($v2,$b) and
getExtentRight($v2,$l) and getExtentLeft($v2,$r).
```

After the rewriting, the tags in a fragment match those in the source, which also makes it easier to convert to the native language of the source. The output of the query, $R, is the extents (bounding rectangle) of the desired region.

- **Produce a map overlaying the Parcel Map and Aerial image**

Assume that we have obtained the georeferenced aerial image from the image library, which is aligned with the other GIS layers. Assume that this image is stored in the GIS as the file *"carmelima*ge". We show one of the five independent queries that are sent to the wrapper. Here, the year of the parcel map is fixed.

```
ans1 = construct $M
where
$M:       <mix:map>
<ArcView_Projects>
              <theme name = $n1>$t1
              <extents>
                 <top>$t</>
                 <bottom>$b</>
                 <left>$l</>
                 <right>$r</>
                </extents>
              </theme>
              <theme name = $n2 >$t2</>
              <theme name =$n3 >$t3</>
              <tables>
                <table name=$n4>
                  <col name=$n5>$v1</>
                  <col name=$n6>$v2</>
                </table>
              </tables>
```

```
</ArcView_Projects>
</mix:map>
in http://wrap.gis.url
and ($n1= “sdcity.shp”) and ($t = 3.62482e+006) and ($b
= 3.6225e+006) and ($l = 481477) and ($r = 481477) and
($n2 = “sdparcels95.shp”) and ($n3 = “carmelimage”) and
($n4 = “Attributes of  sdparcels95.shp”) and ($n5 =
“total  assessed value”) and ($v1 > 500000) and ($n6=
“address block”) and display_order($t3,($t2,$t1)).
```

Here the extents from the first theme are used to limit the map produced to the area extracted from the previous query. Also, the response to the query is a map, which is returned by reference as a URL where the image will be available.

4.3 Spatial Equivalence at the Spatial Mediator

Several mismatches may occur when combining the image and GIS sources. The spatial mediator incorporates rules to handle such mismatches and establish spatial equivalence between corresponding entities. It is not our intent to create an exhaustive set of rules for all equivalence conditions that may need to be included for any arbitrary combination of spatial sources. We believe the mediator needs to be extensible and the designer of a specific system will have the responsibility of including new rules. Once a rule is defined and registered, the mediator will have an engine to check the preconditions of the rule and execute the rule. In this section we briefly discuss the architectural extension to accommodate such rules, the structure of an equivalence rule and how it impacts the query evaluation plan.

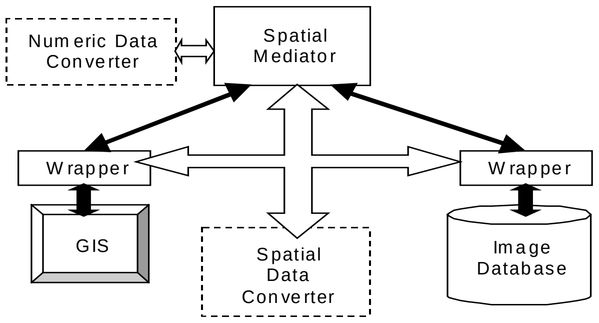


Figure 6. A revised architecture of the system to account for impedance mismatch between information sources

We keep the feature alignment problem out of the scope of this paper. In Figure 6 we show two processes that cooperate with the mediator and wrapper to execute spatial and numeric data conversion. The Spatial Data Converter converts spatial data from one projection system to another. It is controlled by the mediator, but transfers information between the wrappers. The Numeric Data Converter converts between the wrappers and the mediator.

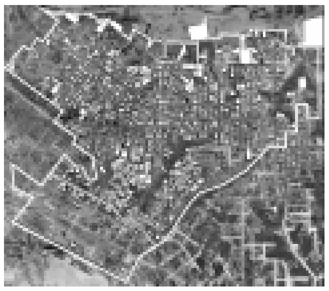
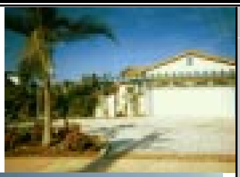
Suppose the GIS layer of the “Police Service Region” and “Parcel Map” layers are in UTM projection, while the image data of the San Diego region is from the USGS 7.5′ quad and is unprojected in geographic coordinates (we call this projection = “geographic” here). This will produce the following changes in the query evaluation steps outlined in Step 9 of Section 4.

Determine Query Execution Plan: The query planner formulates and executes in the following order:

1. First determine the extents of boundary of the region Carmel Valley from the GIS.
2. Check <projection> and <units> tags of the returned result and determine the projection used by the GIS source.
3. Look up rules to find that what image tag or attribute the tag <projection> of GIS source maps to. At this step the mediator discovers that the <projection> tag in the GIS source corresponds to the “projection” attribute of the <mix:image> tag.
4. Look up at the schema of the image sources and finds the values of the attribute “projection”. At this point the mediator finds that all georeferenced images have the “projection” attribute set to “geographic”.
5. Convert coordinates if necessary. For our example, it fires a rule of the form:
6. projection(georaphic, coordX, coordY) :- projection(UTM, coordX1, coordX2), UTM2geo(coordX1, coordX2, coordX, coordY).
7. The predicate geo2UTM invokes the Numeric Data Converter to execute the requisite conversion. As a side effect the mediator uses an image transfer rule and determines that “geo2UTMImage” is the conversion routine for the spatial data.
8. Use the new extents to find the aerial image of the region in the image library.
9. Validate the query that images of the region exist at the requisite resolution.
10. If there are images at multiple resolutions for the region, it uses a rule to chose the finest resolution image covering the entire map region.
11. Initiate transfer of the image from the image library to the GIS. At this point is already known that the image must be transformed before sending it to the GIS system using the “geo2UTMImage” routine. The spatial data converter is the invoked and the converted image is routed to the GIS wrapper. This process was chosen to illustrate the use of the Spatial data converter. In reality, we need to select a cost optimal solution for the conversion.
12. Execute the map retrieval query (which really produces 25 sets of results), and determine the parcels. Since the address block of the parcel will be required in a later part of the query, it is also fetched.
13. Formulate the image query including the sorting and “top 5” instructions.

Although the example here treats a simple case of equivalence management at the spatial mediator, we believe the same architecture will allow us to perform more involved reconciliation between different information sources.

Table 3. Portion of the sample result for the example query (TAV is total assessed value).

Year	**TAV** > $500K		$300K **<TAV** < $500K	
1975				
1980				

4.3.1 Execution of an XMAS query fragment on a GIS

As discussed earlier, portions of the query execution plan which get passed down to the spatial wrapper need to be converted into a query language native to the underlying GIS system. This subsection builds on the running example and sketches a possible solution for a GIS system like ArcView. The underlying principle is that primitive GIS spatial operations can be invoked and composed into larger programs.

A logically equivalent Avenue script is generated whereby primitive spatial operations can be composed to form more complex programs. Let us illustrate this process with the formulation of two consecutive spatial queries: (1) restrict the parcel map to the Carmel Valley neighborhood, (2) using this more focused parcel map, locate all homes whose total assessed value is greater than $500,000.

Essentially here we are querying the underlying GIS system to extract the parcel data that will be overlaid onto an aerial photography. This corresponds to partially filling in one of the cells of the result document for the example query.

- Query (1) can be accomplished by invoking a primitive request called **QueryThemeByTheme**, which takes as input parameters (*ThemeObject*, *SearchTheme*, *spatialRelationship*). In our example query the *ThemeObject* would be the parcel map, the *SearchTheme* would be a boundary theme for Carmel

Valley, and the *spatialRelationship* would constrain the selection using the *"Within"* operator.

- Query (2) can be accomplished by invoking a simple primitive request called **QueryThemeByExpression**, which takes as input parameters (*ThemeObject*, *QueryExpression*). The query expression would be *"[total assessed value] > 500000"*, assuming an attribute field named "total assessed value" in the *ThemeObject*'s associated attribute table.
- The overall GIS query would combine (1) and (2) in the following manner:

```
av.Run("SetEnvironment",
{parcelTheme="Parcel Map", selectionTheme="Carmel Valley Police Service Region",
relation="Within", expression="[total assessed value] > 500000")

av.Run("QueryThemeByTheme", {parcelTheme, selectionTheme, relation } )
av.Run("QueryThemeByExpression", { parcelTheme, expression } )
```

Av.Run is the standard Avenue call to run an Avenue script with arguments from within an Avenue script. The "SetEnvironment" Avenue script takes as arguments a list of attribute/value pairs where the attribute is the name of a variable that will be initialized to the associated value.

- Let us detail the first of these two primitive requests in Avenue. **QueryThemeByTheme** would be written as:

```
Region1 = self.Get(0)
Region2 = self.Get(1)
Relationship = self.Get(2)
Rel = av.Run( "Lookup", Relationship )
Distance = self.Get(3)

If ( Distance = null ) then
        Distance = 0
End

Region1.SelectByTheme( Region2, Rel, Distance, #VTAB_SELTYPE_NEW )

Region1 = av.Run( "SaveSelection" )
```

The "SaveSelection" Avenue script transforms a selection bitmap into a Theme object.

5. Discussion

This paper presented an architecture and a logical schema for Web-based spatial information mediation using XML. We have traced a sample query integrating imagery and GIS sources, through its evaluation at the spatial mediator and dispatching to XML-wrapped geodata sources, where query fragments are translated to the language of the source and executed. We believe that, while a first step in the development of scalable and extensible spatial data mediation systems, this exercise helped us elucidate areas of complications which create the context for future research. These proposed research areas include:

- Development of rules and a cost model for selecting spatial data sources in the spatial mediator. Metadata for each source will describe the source's capabilities, such as size of data, data quality, indexing, available format and projection conversion and alignment routines, the monetary price of retrieving particular data, etc. This information will allow the mediator to estimate which sources need to be queried, in what order, where the retrieved data fragments need to be assembled (i.e. which source should be designated as a "collector" source), etc.
- Development of a general cost model for parsing, evaluating, and distributing queries, and for assembling the results.
- Incorporation of physical integration (alignment) management capabilities in the architecture of the mediator system, and in query planning at the mediator. An "intelligent" alignment mediator will maintain a semantic graph of "alignable layers". This graph will demonstrate, for example, that linework from a soil map and a vegetation map, a vegetation map and a land use map, a coastline map and a political boundaries map should closely align. By contrast, any alignment between a vegetation boundaries and a road network, for example, will have lower alignment priorities, if any. Development of such a semantic graph requires research into persistent landscape features that "show through' multiple geographic layers, and may form the strongest links in the graph.
- Incorporation of data quality issues in the context of information mediation, in particular, inferring the desired accuracy level of geodata sources based on the target query accuracy. The inferred expected accuracy of sources will be used by the spatial mediator in query planning, and become a component of the query cost model.
- Balancing the automated and manual procedures in the process of human interaction with the spatial mediator system. This will be important when we need to put a human in the loop for performing computations in mediated GIS systems.
- Supporting geographic analysis "workflow" as a sequence of spatial queries to the mediator system. This will require preserving intermediate query results, in XML form, at some URLs, so that they can be used as a source for subsequent queries.
- Scalability analysis of the mediation of multiple GIS and imagery sources.
- Supporting alternative mechanisms to associate names with geographic objects. This will address the problem that in general, there may be a many-to-many mapping between the names and geographic objects

References

[1] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, J. D. Ullman: A Query Translation Scheme for Rapid Implementation of Wrappers. DOOD 1995: 161-186. http://www.cse.ucsd.edu/~yannis/papers/querytran.ps

[2] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, A. Yannakopulos: XML-Based Information Mediation with MIX. *Proceedings of the SIGMOD'99* (to appear). http://www.npaci.edu/DICE/Pubs/sigmod-demo99.pdf

[3] P. Buneman, S. B. Davidson, M. F. Fernandez, and D. Suciu. Adding Structure to Unstructured Data. In 6th Intl. Conference on Database Theory (ICDT), LNCS 1186, pp. 336-350, Delphi, Greece, 1997. Springer.

[4] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your Mediators Need Data Conversion! In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, pp. 177-188, 1998.

[5] M. J. Carey, L. M. Haas, V. Maganty, and J. H. Williams. PESTO: An Integrated Query/Browser for Object Databases. In Intl. Conference on Very Large Data Bases (VLDB), pp. 203-214, 1996.

[6] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. STRUDEL: A Web-site Management System. In ACM Intl. Conference on Management of Data (SIGMOD), pp. 549-552, 1997.

[7] B. Ludascher, R. Himmer oder, G. Lausen, W. May, and C. Schlepphorst. Managing Semistructured Data with FLORID: A Deductive Object-Oriented Perspective. Information Systems, 23(8), 1998.

[8] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object Fusion in Mediator Systems. In Intl. Conf. on Very Large Data Bases (VLDB), 1996.

[9] Y. Papakonstantinou and P. Velikhov. Enhancing Semistructured Data Mediators with Document Type Definitions. In Intl. Conference on Data Engineering (ICDE), Syndey, Australia, 1999 (to appear).

[10] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38-49, 1992.

[11] P. M. Mather, Map-image registration accuracy using least-squares polynomials. *Int. J. Geographical Information Science* 9:543-554, 1995.

[12] XML-QL: A Query Language for XML. W3C note, http://www.w3.org/TR/NOTE-xml-ql, 1998.

[13] A. Voisard and H. Schweppe. Abstraction and Decomposition in Interoperable GIS. *International Journal of Geographic Information Science*12(4), Taylor and Francis, June 1998.

[14] A. Voisard and M. Juergens. Geographic Information Extraction: Querying or Quarrying? In *Interoperating Geographic Information Systems*, M. Goodchild, M. Egenhofer, R. Fegeas and C. Kottman (Eds.), Kluwer Academic Publishers, New York, 1999.

[15] K. Stock and D. Pullar. Identifying Semantically Similar Elements in Heterogeneous Spatial Databases using Predicate Logic Expression. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999.

[16] D. J. Abel, Towards integrated geographical information processing. *Int. J. Geographical Information Science* 12:353-371, 1998.

[17] H. Kemppainen. Designing a Mediator for Managing Relationships between Distributed Objects. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999.

[18] S. Shimada and H. Fukui. Geospatial Mediator Functions and Container-based Fast Transfer Interface in SI3CO Test-Bed. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999.

[19] T. DeVogele, C. Parent and S. Spaccapietra, On spatial database integration. *Int. J. Geographical Information Science* 12:335-352, 1998.

[20] W. Y. Ma, " NETRA: A Toolbox for Navigating Large Image Databases," Ph.D. Dissertation, Dept. of Electrical and Computer Engineering, University of California at Santa Barbara, June 1997.

[21] A. Gupta, S. Santini, and R. Jain, "In Search of Information in Visual Media", *Communications of the ACM*, December 1997, vol. 40 (No. 12): 35:42.

[22] A. Levy, A. Rajaraman and J. Ordille: Querying Heterogeneous Information Sources Using Sources Descriptions. *Proceedings of VLDB*, 1996: 251-262.

[23] A. Tomasic, L. Raschid, P. Valduriez: Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 5, 1998: 808-823.

[24] S. Shimada, H. Fukui: Geospatial Mediator Functions and Container-based Fast Transfer Interface in SI^{3}CO Test-Bed. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999, pp. 265-276.

[25] Y.A. Bishr, H. Pundt, C. Rüther: Proceeding on the Road of Semantic Interoperability - Design of a Semantic Mapper Based on a Case Study from Transportation. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999, pp. 203-215.

[26] G. Camara, R. Thome, U. Freitas, A.M.V. Monteiro: Interoperability In Practice: Problems in Semantic Conversion from Current Technology to OpenGIS. In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999, pp 129-138.

[27] C.B. Cranston, F. Brabec, G.R. Hjaltason, D. Nebert, H. Samet: Interoperability via the Addition of a Server Interface to a Spatial Database: Implementation Experiences with OpenMap, In Proc. Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Andrej Vckovski, Kurt E. Brassel, Hans-Jörg Schek (Eds.), Zurich, Switzerland, March 10-12, 1999, pp. 115-128.

[28] Y. Bishr: Overcoming the semantic and other barriers to GIS interoperability. *Int. J. Geographical Information Science* 12, 1998: 299-314.

[29] R. Laurini: Spatial multi-database topological continuity and indexing: a step towards seamless GIS data interoperability. *Int. J. Geographical Information Science* 12, 1998: 373-402.

[30] Geographic Data Exchange Standards: http://www2.echo.lu/oii/en/gis.html

[31] V. Christophides, M. Scholl and A.-M. Vercoustre: Querying Heterogeneous Semi-Structured Data. ERCIM News No. 33 – April 1988. http://www.ercim.org/publication/Ercim_News/enw33/scholl.html