Integrating GIS and Imagery through XML-based Information Mediation

Amarnath Gupta, Richard Marciano, Ilya Zaslavsky, Chaitan Baru
{gupta,marciano,zaslavsk,baru}@sdsc.edu
UC San Diego / San Diego Supercomputer Center

1 Introduction

In recent years the problem of integrating information from heterogeneous information sources has become an active area of research in the database community. These efforts have primarily concentrated on information sources such as web documents, relational databases and text files. In the domain of spatial information systems, researchers have explored system integration architectures, issues and problems in semantic interoperability of spatial information and information integration concepts for spatial data [11,16, 17]. In particular, some recent researchers [14, 15, 19] use information integration methodology from the database community toward spatial information systems. In this paper we investigate a mediation-based approach (referred to as lazy evaluation in [14]) to integrating information from a spatial information system such as GIS, and a database of geo-referenced imagery that possibly lives in a digital library. As in [14,19], our purpose is to provide a user the ability to issue a single query that involves both information sources, and receive a result that combines information from these sources in a seamless manner. In the same spirit we would like to provide an authorized user the ability to make permissible updates to a source by using conditions involving both sources. The goal of this paper is to demonstrate how a mediated system is designed and to step through the query evaluation procedure in an such a system. We must emphasize that our notion of integration 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. We aim to attain "logical integration" by creating correspondences between related spatial information much in the same spirit as mediation systems [10] do for databases. In this endeavor we do show existing physical integration techniques fits into the fold of our information association methodology, but development of such methods is secondary to the focus of this paper.

1.1 Background and related work

Various interoperability approaches and architectures have been discussed in the context of distributed geographic processing and spatial data integration in the last several years. From ad-hoc project based spatial data integration via numerous and never quite sufficient data conversion procedures, with an assembly of a centralized data store, the research focus has shifted to standardization of data exchange, and to the development of specialized software systems to support data interoperability. Useful reviews of GIS interoperability and integration efforts are given in [16, 19, 28, 29]

The standardization efforts in a number of countries resulted in the development of presumably universal standards for data exchange, including SDTS (U.S.), FEIV (France), ALK (Germany), SAIF (Canada), further described in[30]), etc. Coexisting with de-facto commercial spatial data interchange standards (such as ESRI's .E00 files and shapefiles, MapInfo's MIF/MID, AutoCAD's .dxf, etc.), these standards have not yet significantly affected the wealth of legacy spatial information. At the same time, they were not connected with the emerging standards for Web-based data interchange such as SGML and XML.

Among approaches focused on software development to support spatial data interoperability, we can distinguish the following (focusing on those relevant to the architecture adopted in this paper):

- **Cataloguing geographic sources**, or any sources/datasets with locational identifiers. The Alexandria DL, supporting spatial range queries to a variety of resources, is an example of this approach;
- **Developing gateways between databases**, by defining universal schemes and persistent views over a variety of data sources .
- **Federated spatial databases** (also called multi-databases). Based on the client-server model, with simple CORBA or COM middleware connecting the two layers, this architecture supports homogenous views (a common data model) over heterogeneous data sources. Sometimes also referred to as Data Warehousing, or "eager approach" to data integration [14], this approach proved efficient for relatively small number of sources with known structure. 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]

- **Mediator-based systems**. The three-layer architecture of such systems includes foundation layer (databases with wrappers), mediation layer (supporting exchange of queries and results between wrapped legacy data sources and applications), and application layer (user interface) [10]. The main advantage of this architecture is its modularity and scalability, and the ability to use sources with no structure or implicit structure. Derived from such mediator database systems as TSIMMIS [1,8], DISCO [23], and Information Manifold [22], these systems support combinations of query results from individual sources rather than combining the data. Among the recent examples of this approach are the extensions of INRIA's Aquarelle project, and research described in [24] and [25]. In the first prototype (http://cosmos.inria.fr:8080/poql.html), a set of a priori unknown web servers, with varying structure can be queried using a language called POQL. It supports Web-based querying of multiple servers where both structure and data are queried at the same time. 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]. Semantic heterogeneity between different representations of transportation networks and the ways to overcome it within a mediator-based system are addressed in [25]. The proposed semantic mapper maintains a library of well-known legacy interfaces and a domain ontology, and serves as a middleware dealing with wrapped legacy data sources. The prototype wrapper architecture developed in [24] uses the VisiBroker as the middleware supporting IDL to Java mapping, and is implemented on CORBA 2.0.

Hybrid approaches combining features of the above architectures, have also been proposed. For example, a 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 Integration – an information systems scenario

From a database researcher's viewpoint the task of information integration translates to establishing logical equivalence relationships between different pieces of data. Suppose A and B are two information sources and we would like to is to *logically* create a composite information source C, by piecing together information elements from A and B. Once we have constructed this logically integrated source C, we may now pose the following query Q: *construct a virtual document having the title "TITLE" and in the body put a 2×2 table T, as formulated in Table 1*. Note that each entry specifies how it will be filled up from the integrated information source C.

Table 1. The result T on the integrated source C.

	Column 1	Column 2
Row 1	child of c5 > 9	element after c9
Row 2	avg (all leaf nodes)	subtree rooted at c4

The problem is, however that the raw sources A and B store, organize and retrieve information in completely different ways. As shown in Figure 1, source A may be a web document and organizes its information elements as a tree and source B may be a proprietary system that organizes its information elements like a graph. The question in information integration how can we logically construct C from the

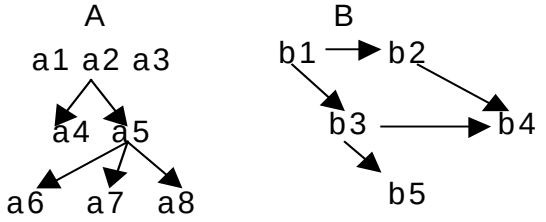


Figure 1. A is a tree-structured information source and B is a graph-structured information source.

raw sources such that the query can be answered. We call C *logical*, because C may not be physically created as a separate repository. The information mediation approach is to define rules on the raw sources such that the elements of C may be defined as a logical association between elements of A and B. For example, the rules may be:

- equate the element a3 with the element b1
- b5 or any of its descendants will not be included in C
- a child of a5 whose value is greater than 9 should also be the child of b4.

The resulting "integrated" information (after renaming) may look like Figure 2.

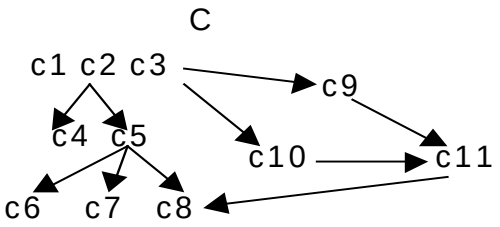


Figure 2. In the integrated source C, elements and relationships from sources A and B have been put together, subject to some association rules

This example shows three important aspects of an information integration scenario:

- not all elements of the raw information sources (such as b5) may be "exported" to the integrated information source. Thus the integrated source only sees a view of the raw source.
- information association is executed using a set of rules.
- a query may restructure the original data to a form (a table in the example) not present in the original or integrated sources (tree or graph)

1.3 Integration – the spatial information scenario

This paper proposes an XML-based solution to the problem of spatial information integration. In this section we introduce our notion of spatial information integration through a fictitious motivating example. Unlike the database example in the previous section, spatial information will necessarily have typed information and methods defined on them. This implies that the user queries can contain these exported methods, which will also play a role in decomposing the query because not all operations can be done on all types and hence on all sources. Consider two independent, autonomous sources S_1 and S_2. S_1 is a GIS containing themes such as the soil map, parcel map, digital elevation map and transportation network map of Southern California, and S_2 is an image library that has satellite images, aerial images and property photographs of different regions of Southern California. For our example, let us assume the image library is managed through a DBMS, which, in addition to metadata (which include timestamps and have been georeferenced for our discussion), can serve a complete or a cropped version of any image. The goal of creating the integrated spatial information source S_3 is not to pull layers of S_1 and images from S_2 into a third monolithic system, but to maintain in S_3 a logical representation of the information in S_1 and S_2, tied together by equivalence relations. Regardless of how themes are structured within S_1, we initially assume that logically S_1 can be represented in S_3 as a tree-structured source (like an R-tree, for example), where a node of the tree represents the extent of a theme, with additional metadata describing the properties and content of the theme. For the source S_2 let us initially assume that the images are associated with metadata and may additionally have been post-processed by a classification process, a segmentation process, or an

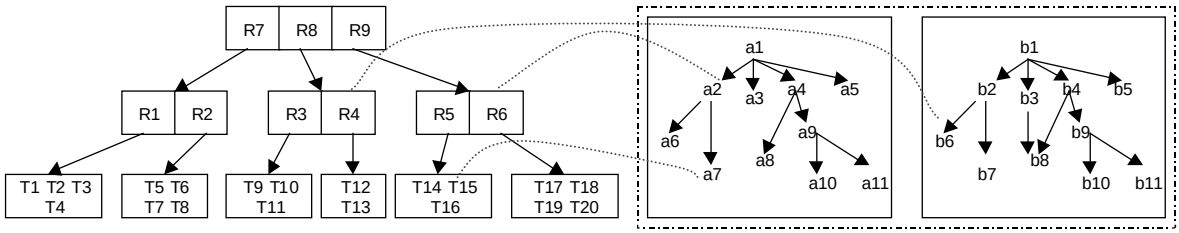


Figure 3. The structure of the integrated information source. Note the dashed associations between the R-tree-structured representation of the GIS source and the set of image sources.

annotation process. Under this assumption, the view of S_2 exported to S_3 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 segment represented by n_1 is contained within the segment represented by n_2.
The structure of S_3 then is that of a graph, where the nodes of tree-structured views of S_1 and S_2 are connected through a number of equivalence relations. These relations may be established by rules that may specify inter-object associations by methods like:

- containment conditions, such as the extent of image object node n_3 in S_2 is covered by theme node n_5 in S_1
- explicit spatial or temporal join conditions
- loose logical associations, such as both items refer to the city of San Diego

When the integrated source S_3 is thus defined queries and update requests can be posed against it.

1.3.1 An example query

In this paper we take the viewpoint that the result of such a query is always a document that may contain text, tables, figures, images, vector graphics and maps. We will use the following example query based on the sources S_1 and S_2 from the previous section. "Using the *Assessed Total Value* (land price + improvement estimate) of the parcel maps of San Diego, specified as *Carmel Valley* in the 1998 *Police Service Regions* of San Diego, a set of aerial imagery of the regions and photographs of house properties in the region, produce a document with the following table (Table 2)". The task here is:

- first find the region corresponding to "Carmel Valley" from the Police Service Regions of 1998
- for each specified year classify this region into the five land price brackets (as shown), and generate one map for each bracket
- for each single-bracket map from the previous step overlay it on the appropriate aerial image of Carmel Valley
- for each subregion, also find house properties in the region, rank them by price and choose the top five photos
- arrange the information as shown

Table 2. The output of a complex query (TAV stands for Total Assessed Value)

Year	**TAV** > $500K		$300K < **TAV** < $500K	$200K <**Land price** < $300K	$100K <**TAV** < $200K	**TAV** < $100K
1975	Join TAV map of qualified parcels with aerial photo of the same regions	Property pictures of 5 most expensive properties in the same regions	Same conditions as in Column 1			
1980						
1985	Same conditions as for Year 1975					
1990						
1995						

The query produces a table of 25 maps and 25 sets of house photographs -- a representation to show urban change in San Diego. We intentionally chose a tabular presentation of the information to illustrate that the integrated system may not always produce a single map as its output. It also demonstrates the need for the mediator, which decomposes a single query into multiple simpler queries to be executed by the GIS. We use both physical integration (the images from the library are fetched to and overlaid onto the TAV map) and logical integration exercised through the join conditions on the aerial images and maps, and on the join conditions of the property picture addresses and the qualified regions in the GIS.

In the rest of the paper we describe an architecture that achieves the spatial integration task outlined above. Our primary contribution is to propose a logical methodology to reduce the impedance mismatch between existing spatial systems through XML, the emerging information interchange standard.

2 Information Mediation using XML

2.1 Background

The approach we described in the previous section achieves information integration by using a two-part middleware between the information sources and the end-users' application. In this approach the user application interacts with a large number of disparate information sources through a single interface called the *mediator* (the first of the two-part middleware). The mediator accepts a user request, breaks up the request into small fragments according to the capabilities of the sources and delegates the request-fragments to the appropriate sources. As an example of capabilities, a GIS can perform spatial operations on theme objects, while an image database can perform similarity-based operations on images or image segments. When the sources process the request and return the results, the mediator integrates the results and sends the combined information back to the user.

The mediator typically communicates with a large variety of heterogeneous information sources. To manage the diversity of protocols for each source, rather than directly communicating with the raw sources, it communicates with a proxy of an information source called a *wrapper* (the second part of the middleware). The wrapper acts as a two-way model translation device: it communicates with an information source in its native language/API, and communicates with the mediator in a commonly agreed language. The task of the wrapper, therefore, is to translate a request from the mediator's language to that of the information source and transform the results provided by the information source back to the mediator's language. As noted in [1], a wrapper can process *direct*, *logically equivalent* and *indirect queries*. A direct query is a request that can be satisfied by a primitive operation provided by the underlying information source. The query "Find all houses in census tract 06073001 having median income greater than $50,000 by census block group" can be directly mapped to a single request, and is an example of a direct query. A logically equivalent query is a query for which the underlying system does not provide a single operation. A query like "For each census tract in San Diego that has over 30% minority population, find the zip code boundaries that intersect with it" may not be directly translated to a single GIS request. However, a script can be generated to produce a result that is logically equivalent to the query. In this case, the wrapper's task is complicated by the fact that it has to *compose* a program from smaller modules to produce the result. An indirect query is a query that is not supported by the underlying information source, but the wrapper has the computational capability to answer it. For example, a wrapper may compute the correlation coefficient of a sequence of number pairs retrieved from a database, which the database itself is unable to perform the computation. In the present work we focus mainly on processing direct and logically equivalent queries, and provide a simple example of an indirect query.

In this paper we present the functional architecture of the spatial information mediation system, the representation of the GIS and image database sources at the wrapper and at the mediator, and the mediation logic to illustrate how the integration between digital imagery and GIS can be achieved.

2.2 The MIX framework

Our information integration framework is called MIX (***M****ediation of* ***I****nformation using* ***X****ML*), where:

- Each source exports information as XML. 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 are obligated to produce documents that conform to an associated DTD. As we will show in the following sections, 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. We have developed a query language called XMAS [2], which builds upon ideas of languages like 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. An effect of the grouping operation is that depending on the query we may arrange the same information in different ways. In our running example, the time line is produced by grouping the spatial data and images under their timestamps. We also illustrate in this paper how we have modified the original XMAS language proposed in [2] has been "specialized" by using a reserved namespace (i.e., set of tag names) for spatial objects.
- The query evaluation and integration process will 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. In our example, assume that the response of the query contains 50 pictures of property information for homes in Carmel Valley in the 1940s. The user may decide to look at the textual metadata of the first 20 and then retrieve the images of only three properties in the resultant document.

Figure 4 shows the architecture of the MIX system modified specifically to handle spatial information. In this architecture, the XMAS query from the user application is developed on top of a main mediator. The

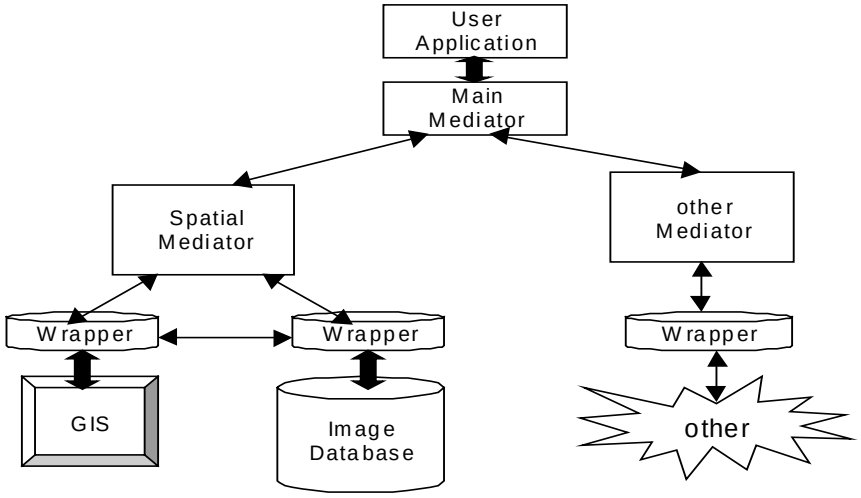


Figure 4. All the links shown in this figure communicate through XML for data and XMS for queries. The inter-wrapper links communicate by exchanging binary information if needed.

XMAS version of the example query is given in Figure 5. Instead of explaining each step of the query let us point out a few key elements.

- The query is directed to the mediator without specifying the location of any source. The only use of the reserved namespace tag mix:source is to indicate which definition of "Carmel Valley" should be used.
- Wherever we use the notation mix:tagname we refer to the fact that the mix mediator is aware of the type of data and has specific ways of handling it. For example, mix:region will not be expected to have a "length" attribute.
- We use the function category(price,totalValueCategory) as a separate function that examines the total assessed value of a parcel in the parcel map and assigns the parcel to the correct bracket.
- The query defines a table by first grouping the results by year (notation {$y}) and then by total assessed value category (notation {$c} within the {$y} group).

- Wherever the same variable (e.g., $r) is used multiple times, it has to bind to the same constant. Thus the aerial image and the map object must be of the same region, and the address of the property must belong to the parcels satisfying the category condition.
- The spatial predicate within(region1,region2) 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.
- The predicate display_order(mapData1, mapData2) 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)).

In general, the application mediator will receive a query containing both non-spatial and components. It will detect the spatial portion of the larger query and assign it to the spatial mediator for evaluation. In this paper we do not give the details of how this may be achieved in general. We assume that the application mediator employs a set of rules to determine that the query needs to be handled by the spatial mediator. The rules can be very simple. In our example, the fact that the formulation of the query requests maps and images is sufficient to direct it to the spatial mediator. We do not consider the issue of multiple GIS sources in this paper. However, we take the position that the basic methodology described here will not be different even for multiple GIS sources.

In the next two sections we discuss the functions of the mediator and the wrapper in the light of the example.

```
answer = construct $A
where
$A:<table>
        <row>
            <year>$y</>
                <totalValueCategory>$c
                    <totalValue>$tv</>
                    <mapColumn>
                        <mix:map>
                            <mix:region mix:source=$s1>$r
                                <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>$y</>
                            </>
                        </>
                    </>
                    <pictureColumn orderby=$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).
```

Figure 5. XMAS version of the example query

3 Query evaluation at the spatial mediator

The task of the spatial mediator is to parse the spatial part of the query and generate an evaluation plan. There are three parts in this process. The first part entails fragmenting the query between information sources and determining the order of execution of each fragment. The second part is to use the knowledge about each source to rewrite the query in a way that the source can evaluate the rewritten query. The essence of information integration is carried out by these two parts. The third part is to send out the rewritten query fragments to the sources, collect the result fragments from each source and pass them on to the application mediator. We next inspect how these steps are executed.

3.1 Query planning at the mediator

In this section we walk through the states executed by the mediator – note that the process of integration manifests itself by an interdependent sequence of subqueries between the two sources. Naturally, the steps indicated here will be different for another query. However it is representative of the query planning in many typical queries.

1) **Determine Map Request**: The tag mix:map dictates that the mediator needs to create a map. It 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 specifies the region to be mapped and informs that this region can be found in the source having a theme “Police Service Region” within the provenance of “San Diego”. This information is assumed to be in the GIS them in this paper. If this is not available, the system will look for alternatives like finding the information from a geographic name server. The mediator searches the DTD exported by the GIS wrapper and locates 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**: Next it determines the condition that the mix:regionName tag needs to have the value “Carmel Valley”. It also consults the DTD and maps the tag regionName to the field name srvRgn in the “Police Service Region” table. Hence the mediator places query condition *srvRgn= “Carmel Valley”* as part of the query fragment to be passed on to the GIS wrapper.
4) **Identify Map Attribute 1**: The next tag mix:mapData 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 (and hence themes) are requested in the query, 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 1**: The mediator inspects the conditions associated with “total assessed value” and finds that it has to satisfy the function category(price,totalValueCategory) and also observes that the results are grouped into columns based

on its category value. It expands the function to determine that there are 5 possible categories each parcel map data may belong to. It infers that since for each category a map needs to be produced, it creates 5 correlated queries to the wrapper for each independent subquery. It generates the corresponding query conditions for the GIS wrapper.

6) **Identify Map Attribute 2**: The next mix:mapData tag specifies the next element to be mapped. The mediator determines that this item is an aerial map and belongs to the image library. It also determines that it needs to satisfy query conditions on the year, the resolution and the georeference to identify images for the query. But although the mediator has the DTD produced by the image wrapper, whether the query can be safely answered is unspecified at this point because the region specifying the georeference is not computed yet. Hence the mediator forms a partial query for the image wrapper.
7) **Formulate Query Fragments for Wrapper**: The mediator observes the end of the mix:map tag and produces one independent and several dependent query fragments for the wrappers.
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) has to be performed in two steps: first getting the address block from the Parcel Map table in the GIS source, and then formulating a range query on the street number in the image library.
9) **Determine Query Execution Plan**: The query planner formulates and executes in the following order:
    a) First determine the extents of boundary of the region Carmel Valley.
    b) Use the extents of to find the aerial image of the region in the image library.
    c) Validate the query that images of the region exist at the requisite resolution.
    d) 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.
    e) Initiate transfer of the image from the image library to the GIS[1].
    f) 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.
    g) Formulate the image query including the sorting and “top 5” instructions.

3.2 Examples of a rewritten query fragment

As a result of the query planning process, query fragments are sent to the wrapper in the form of XMAS. However, this rewritten query uses the table and field structure as well as the 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 = $n1>
    </theme>
    <tables>
        <table name=$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>
```

[1] We assume for this work that the image and the GIS layer can be aligned.

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).

Note that after rewriting the tags are now specific to the source, and this makes the query easily convertible to a script in the native language of the source. We would also point out that the output of the XMAS query is only the extents (bounding rectangle) of the desired region.

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

Let us assume we have already obtained the aerial image from the image library and that by virtue of the fact that it is georeferenced, it is aligned with the other GIS layers. We assume that this image is stored in the GIS as the file *"carmelima*ge". We will show one of the five independent queries sent to the wrapper. In this query 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.

3.3 Spatial Equivalence at the Spatial Mediator

The query planning process described in the previous section did not consider any mismatches between the two information sources. In real life we are more likely to witness several mismatch problems – the projection on the image and the GIS layer may not match, the coordinate systems may be somewhat different, and the features may not align. Unlike the application mediator, which has no domain knowledge, the spatial mediator is designed to incorporate special 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 putting in any 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.

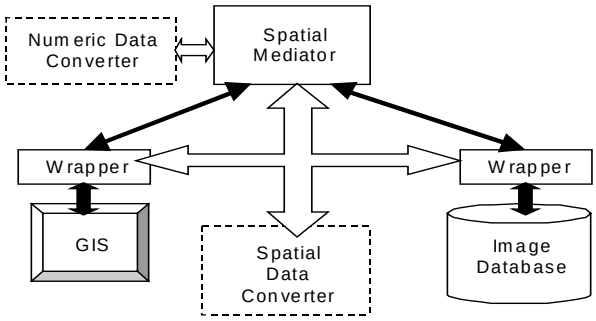


Figure 6. A revised architecture of the system to account for impedance mismatch between information sources

We keep the feature alignment problem out of the scope of this paper. In Figure 6 we show two computation agents that cooperate with the mediator and wrapper to execute spatial and numeric data conversion. The spatial data conversion agent performs operations such as converting a spatial data from one projection system to another, while the numeric data conversion agent performs information conversion that are much smaller in scale. More importantly, the spatial data converter is controlled by the mediator, but transfer information between the wrappers, while the numeric data converter helps the negotiation process between the wrappers and the mediator.

Consider that 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 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 3.1.
Determine Query Execution Plan: The query planner formulates and executes in the following order:

a) First determine the extents of boundary of the region Carmel Valley from the GIS.
b) Check <projection> and <units> tags of the returned result and determine the projection used by the GIS source.
c) 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.
d) 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".
e) Convert coordinates if necessary. For our example, it fires a rule of the form:
projection(georaphic, coordX, coordY) :- projection(UTM, coordX1, coordX2), UTM2geo(coordX1, coordX2, coordX, coordY).
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.
f) Use the new extents to find the aerial image of the region in the image library.
g) Validate the query that images of the region exist at the requisite resolution.
h) 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.
i) 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"

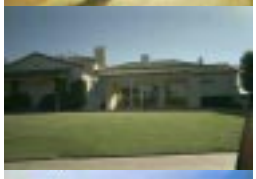
routine. The spatial data converter is the invoked and the converted image is routed to the GIS wrapper. This process was chosen to illustrate the use of the Spatial data converter. In reality, we need to select a cost optimal solution for the conversion.

j) 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.

k) 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		

network flow optimization), and a presentation source (e.g., surface generation and mosaic creation) all in one, but is not equipped with a generic query language for declarative access. Our GIS wrapper overcomes this problem by maintaining an internal model of the GIS source, including both the schema information and the instance information. The instance information is populated only when a query is posed. The wrapper performs model translation by first transforming the result returned by the GIS into this internal model, and then exporting it as an XML document.

4.1.1 The internal schema of the GIS wrapper

The motivation behind having a lightweight internal data model for a GIS wrapper is as follows. Although we need to redesign a wrapper for each different GIS product, certain parts of the wrapper are going to be common across all products. For example, most GIS sources will recognize map layers (coverages, themes, etc.) and a large body of common operations such as overlay and spatial intersections. These ubiquitous objects and functions can be considered as a very 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.

For example, in ArcView's Avenue the object returning function *within*(region1, region2), where region1 is a theme object and region2 is region object, is called #FTAB_RELTYPE_ISCOMPLETELYWITHIN, and needs to be invoked as:

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

To MapInfo's MapBasic syntax, the same function could be mapped as

SELECT * FROM Region1 WHERE OBJ ENTIRELYWITHIN Region2.OBJ

As a consequence, when the GIS source returns the results of a query, the results 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 in terms of a simple type system, in order to encapsulate what a mediator needs to know about the source in order to effectively evaluate queries directed to it. With a **collectionObject** at the top level, it further distinguishes between a **themeObject** and a **dataObject**. 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

Note that these types are not defined with the usual rigor found in spatial information systems, because they are not created for performing geometric operations in the wrapper — execution of such operations is the

responsibility of the GIS underneath. The purpose of these types is to translate 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. During the creation of a wrapper for a specific GIS system, these types can be specialized by single inheritance. Hence the designer developing the wrapper for a specific GIS, can define a transportation network as a networkObject. For every built-in or user-extended type a list of function signatures is also defined. For example, for curveObject spatial intersection is defined as: intersect(curveObject, regionObject) and so on. Note that we did not care about issues like a region intersecting a curve will generate a set of point objects, although a **set** is not included in the type system. We contend that regardless of the semantics of the geometry, a wrapped GIS source will eventually produce a themeMap object and an associated table, or return an empty result. The only reason we need the function signature is that the predicates specified therein will be made visible to the mediator[2] and will be used to formulate valid XMAS queries. In addition, this allows 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, without worrying about the precise semantics of the operation.

In the rest of the paper we assume that ArcView from ESRI is the underlying GIS and illustrate both catalog service and query translation.

4.1.2 Catalog 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.

```
<!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>
```

Based upon our experience with ArcView, we distinguish between a base theme set **B** and view theme set **V** in any GIS instance. A specific 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

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

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. Recognizing a theme as a view theme can be a non-trivial task. We have taken a simplified approach to the problem. We assume that the projects in the system comprise universe of themes[3]. 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. The dependency graph is maintained by the wrapper for later use. 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 2. 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 time the temporal order may be implemented through a data structure like the interval tree.

4.1.3 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.

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

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.

4.2 Wrapping an image Library

Our intent in wrapping image libraries is to provide a uniform interface to access complete or partial digital images, and image features computed from images in by an image processing or pattern recognition method. The uniform access also includes any metadata associated with a singleton image or image collection. In our experience most database management systems (such as Oracle, Informix and DB2) that support access to imagery export only limited query capability on image features and permits little manipulation (such as cropping) or analysis (such as feature extraction or classification) operations on images. Our internal model for an image source is based on the vision of information sources with both metadata retrieval and content-based retrieval abilities – we have created such sources [21] using customizable image analysis engines (e.g., QBIC from IBM, Virage Engine from Virage, Inc.) in conjunction with database management systems.

4.2.1 The internal model for image sources

We model the image wrapper as a middleware that recognizes the following data types:

- **image**: an image is associated with a set of standard metadata like its dimensions, format and pixel depth (bits per pixel). An image is also modeled to be multiband, and can be retrieved a band at a time or up to three bands together. The user can also perform standard operations like cropping, rotations, change in brightness and contrast on images. 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. Since our image model is general, we allow an image library to provide similarity functions based on features (e.g., it can request all images having some segment with texture similar to this [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.

4.2.2 Examples of wrapped image content

Figure 7 shows an aerial image, the segments produced by the NETRA [20] segmentation procedure on the image, and an enlarged portion of the segment boundaries respectively. This can be exported to the mediator using a format presented by the following XML fragment.

```
<mix:image location= “url1” source= “USGS 7.5 minute quad” projection= “geographic”
georeference_units = “degrees”>
<date>1995</>
<mix:image:group>
    <mix:image:children>
        <mix:image:shape upperLeftX= “..” upperLeftY= “..” lowerRightX = “..” lowerRightY = “..”>
            <mix:image:geometry> Pointset
                <url>url2</>
            </>
            <mix:image:features>
                <mix:image:texture>
                    <url>url3</>
                </>
        </>
        <mix:image:shape> … </>
    </mix:image:group>
</mix:image>
```

In our adaptation of the MPEG4/VRML node types into the mix:image tag family we have made some changes to incorporate feature vectors, which are treated here like the appearance node type in standards and adopted the modification that data can be placed inline as well as can be referenced through a URL.

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