AMUN: An Object Oriented Model for Cooperative Spatial Information Systems

Eric LECLERCQ, Djamal BENSLIMANE, KokouYÉTONGNON
LE2I - Équipe Ingénierie Informatique - Université de Bourgogne
9, rue Alain Savary - BP 400
21011 DIJON Cedex - FRANCE
e-mail : {Eric.Leclercq,Djamal.Benslimane,Kokou.Yetongnon}@u-bourgogne.fr

Abstract

The diversity of spatial information systems promote the need to integrate heterogeneous spatial or geographic information systems (GIS) in a cooperative environment. We present an on going research project, called ISIS (Interoperable Spatial Information System), which aims to build an environment to support interoperability of GIS by interconnecting spatial data repositories and spatial processing resources. Our solution combines techniques from traditional interoperable information systems, spatial data modeling and distributed object oriented databases. While object oriented data modeling impact has been studied in spatial databases, research in model for distribution of both data and operations has received little attention. We focus in this paper on modeling problems and describe an hybrid spatial functional OO model that provides tools to model distribution and to resolve heterogeneities.

1 Introduction

In recent years, a significant amount of research have been directed towards spatial database management systems. They have raised several problems including spatial data modeling, spatial data structure implementations, spatial query languages, query processing and systems architectures [3, 4, 10, 14]. On the other hand, we are witnessing the rapid development of high speed world-wide networks which provides access to a large diversity of informations sources, particularly spatial information systems.

This has created the need to integrate heterogeneous spatial information systems in a cooperative environment. Existing spatial information systems are often designed for specific range of applications, each system may have its own spatial data model and spatial processing capabilities. Users would like to query and manipulate these systems in a uniform way, but the diversity of models and geoprocessing functions specificities render the design of integrated or cooperative GIS a very difficult task to carry out.

Traditionaly, research effort in cooperative information system has been directed at several issues including:

- the definition of concepts and model for allowing uniform access to multiple heterogeneous data sources. Sheth and Larson in [11] present a survey of solution for federating traditional heterogeneous databases systems. Many approaches focus on the definition of techniques for resolving semantic heterogeneity and for constructing integrated schema to support interoperability of autonomous information systems. An alternative to federated systems based on global integrated schema is the multidatabase language approach [9]. It avoids the problems inherent to the creation of global integrated schemas by using a multidatabase language to support interoperability;

- the definition of concepts and tools for searching, discovering and retrieving informations distributed over a network. New architectures and paradigms are being proposed to enable designers to take advantage of existing networking technology while designing distributed computing systems. They range from client-server architecture to distributed object based processing environment such as CORBA and Web based systems. In client-server architecture, client objects communicate with server objects to request data or processing functions. Distributed object based environment is a generalized client-server paradigm in which a cooperation bus is used to allow computation across different platforms. An intermediate model, typically an OO model, may be used to wrap the components of the different platforms, thus providing them with a uniform interface.

In this paper we address issues relevant to GIS interoperability. We present ISIS (interoperable spatial information system) to allow users to query and manipulate these systems in a uniform way. ISIS is a first step to a cooperative environment. Issues that must be dealt with in order to address GIS interoperability include: 1) distributed access to multiple heterogeneous data sources, 2) data model heterogeneity [15], 3)schema and query translation between components. The key characteristic of the project are:

- a mediator/wrapper oriented architecture which provides to users a uniform access to distributed spatial databases;
- a distributed spatial OO data model (AMUN) which provides a uniform design of spatial data stored in differ-

ent GIS and handles localization and distribution of spatial objects;

• an implementation build on the top of a distributed object architecture like CORBA which provides a good framework for the definition of a multilayer set of abstract and concrete services that are mapped to mediators and wrapper. This allow systems to share spatial data and geoprocessing functions.

The remainder of the paper is organized as follows. First, we summarize related approaches to GIS interoperability, second we describe how to use mediators and wrappers to provide interoperability between different GIS. Third, we introduce the distributed spatial object model called AMUN. Fourth, we present an overview of the ISIS environment and its operational architecture. Finally, we conclude essentially with a summary of the paper.

2 Related work

Recently, several researchers have focused on interoperability of GIS [1, 7, 5, 13]. Ken Gardels [5] discusses the fundamental requirements of GIS interoperability: 1) a generic model to support a variety of GIS functions and capabilities, 2) a set of known tools or functions necessary for solving problems and 3) heterogeneous methods for exploring and accessing spatial information and resources in a network of systems.

Yaser Bishr et al [1] describe different levels of GIS interoperability:

• Platform level: is devoted to hardware and network protocols interoperability. It provides support for transferring data files between systems. Users need to have prior knowledge of remote file formats and invoke the appropriate converters on the transferred files.

• Syntactic level: is concerned with data and systems (GIS) interoperability. It allows physical data exchange between systems. Traditional solutions to GIS interoperability are defined at this level. They are based on format conversion. Software tools are defined for structure conversion between pairs of GIS, by typically using a common exchange format to produce an intermediate structure which is later mapped to the final structure. The identification of remote file formats and the required conversions are transparent to the users. Other solutions at this level, allow users to connect to a remote GIS and to query the remote system using their own local language. This solution is comparable to the multidatabase language approach of traditional databases.

• Application level: is relevant to syntactic and application semantics (data model) interoperability. Interoperation at this level is based on a global virtual view of components GIS. Users queries are formulated on this virtual view and mapped on the remote GIS. This solution is similar to federated database approach based on global integrated schema.

Different approaches have been proposed for achieving the above requirements and levels of interoperability. They can be categorized as follows:

• the federation approach, which uses a global integrated schema to represent objects contained in component GIS. First, the spatial schema of the participating systems are mapped into a common data model. Next, semantic conflicts among the schema are detected. And finally, correspondence rules are used to integrate the translated schemas to create global federated schemas;

• the persistent object approach in which each GIS or data repository is seen as a persistent store for spatial objects described by a common data model [7]. The localization of the spatial objects is transparent to end users. It defines an object environment and related tools to allow reusability of the functionalities of the participating systems;

• the interface approach extends each GIS with an interface which describes data from remote GIS in the local model. Typically an extended application programmer interface (e.g OO interface extended with query processing capabilities) is used as local model. This allows a user to access spatial objects from other systems as local objects. The interface handles query translation and maps query results into local objects.

Recently, the OpenGIS consortium has defined a generic framework and guidelines for applying classical open system approach to geographic information processing [8]. The primary intent of open system model is to allow sharing of data, resources and systems services among applications; to facilitate exchange of information among heterogeneous systems; to enable the reuse of software components; and to permit the design of extensible systems. The OpenGIS specification extends the above properties to geographic information systems 1) to define an operational model in which spatial object or features are not defined by their structural properties, but are rather defined in terms of interfaces and 2) to allow the development of geoprocessing on different data architectures. The OpenGIS guidelines define three interoperation models. The essential model defines geographic formalism such as types, schema and services for representing real world entities [5]. The open geodata model can be used to define behavior or methods for geographic elements and to specify a catalog of meta information and spatial references. Finally, the OpenGIS service model define functions for assembling spatial objects and their interface to complex queries and geoprocessing operations.

Several authors have proposed solutions for GIS interoperability based on the OpenGIS specifications. Agnès Voisard et al [13] propose a multilayer approach to the OpenGIS design problem. Each layer correspond to a different level of abstraction, from application level to data access level. Their decomposition provide a good framework for facil-

itating the design of interoperable GIS. Nittel in [7] uses OpenGIS specifications in a persistence object approach. Their persistent object manager (geoPOM) provides a homogeneous interface to heterogeneous spatial data repositories. geoPOM's data model is composed of two parts: an object oriented data model based on ODMG93 and a set of predefined spatial object types based on OpenGIS specifications.

3 An overview of ISIS's architecture

This section presents the functional architecture of the ISIS project. The primary goal of the project is to achieve interoperability among GIS by allowing the realization of distributed spatial object system. The main goal of the architecture are: 1) to allow and facilitate access to multiple spatial information sources, 2) to describe, compose and customize information from different sources according to a specific application domain, 3) to allow incremental definition of the system by using dynamic link between mediator and wrapper, and composition of interface repository. Figure 1 shows the functional architecture of ISIS. It consists of a network of software components grouped in three logical levels devoted to mediators, wrappers and local GIS.

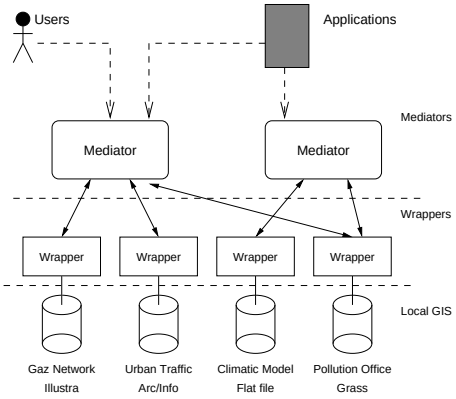


Figure 1: ISIS Mediation Architecture

The mediator level consist of a set of abstract services which are used to provide a uniform and customized view of different heterogeneous data sources. Each view is represented in the AMUN model and is specific to an application domain. This allows the definition of virtual GIS at the user application level. Virtual GIS contains virtual objects which aggregate complex structured objects existing in one or more data sources. Mediator's abstract services handle the decomposition of tasks into subtasks, determine the appropriate data sources and operations required by subtasks and dispatch the subtasks to different local GIS and recombine the results. In summary a mediator comprises an interface repository manager, a distributed object query service, a global transaction manager and an object manager. The object interface repository consists of a set of schemas $m - sdb$ expressed in AMUN common data model. A schema $m - sdb$ corresponds to the description of heterogeneous data (objects classes described by their interface) from one or more sites. The interface repository is associated with an object specification language (ISL). It describes shared spatial object, shared spatial operations and data sources. The distributed object query service is responsible for the decomposition and optimization of global queries. The query decomposition phase must take into account specific geographic operators that are located on different sites. Thus, global decomposition is more complex than in traditional systems. The primary intent is to maximize parallel execution of inter site spatial joins. The global transaction manager is responsible for global concurrency control among tasks. It maintains semantic correctness of global transactions. Global transaction management issues are currently not dealt with in ISIS. The distributed object manager provides tools to create, describe and manage the interface repository.

The wrappers level is a set of concrete services (System GIS Services). The goal of this level is to encapsulate each GIS in a generic spatial object server by a wrapper. A wrapper resolves heterogeneities between a spatial data server and the rest of the system. This is done by defining mappings between local schema (GIS schema) and schema in AMUN. It also deals with invocation of services, execution of operations and data access to and from the underlying spatial data server. A wrapper W_i contains a schema $w - sdb_i$ obtained by translating an export schema $l - sdb_i$ from the local data model to the common data model AMUN, it also describe spatial operations. Contrary to mediator's schemas which combine information from various sources, a wrapper schema $w - sdb_i$ describes a single data source. Furthermore, there is no direct relation between $w - sdb_i$.

The local level consist of a set of local GIS. It is devoted to the definition of a set of export schemas $l - sdb_i$ which correspond to partial views of data sources expressed in local data models.

The operational architecture is presented in section 5, AMUN is presented in the following section.

4 Definition of a common spatial data model

One of the most problem to be dealt with, when interconnecting heterogeneous GIS, is to provide a unique spatial data model and language to access and manipulate the spatial object stored in different repositories. In ISIS environment, we have defined a common spatial data model called AMUN which can used to represent traditional data, spatial data and to manage distributed objects.

As shown in figure 2, AMUN is a three levels model. The first two levels specifies a set of concepts to represent

respectively traditional data and spatial data with attached spatial operations. The last level is composed of new concepts to manage distributed and heterogeneous objects. In the following, we detail the AMUN's concepts defined at each level.

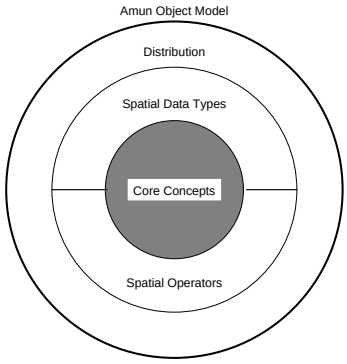


Figure 2: The three different levels of AMUN data model

In this paper we focus on the distribution of spatial data, a description of the concepts for distribution of spatial operation can be found in [6].

4.1 Core concepts

The basic concepts of AMUN are: type, object, class, and the extent of a class. They are described by attributes and operations and their respective signatures. A type interface consists of the properties and operators associated with the type. An interface describes the behavior of the type. Attributes and operations are described by their signatures.

4.1.1 Definitions

The following sets will be used throughout the remainder of this section : O represents the set of real world objects, D is the set of atomic domain values (integer, real, character,...), OID is the set of object identifiers, C is the set of all classes, A is the set of attribute names, OP is the set of operation names.

Definition 1 *(type system T)*
A type system T is defined as follows:
• **integer, string, float, boolean** $\subseteq T$ are atomic types
• if $t_i \in T, i = 1, \ldots, n$ then $tuple(t_1, t_2, \ldots, t_n) \in T$ is a structured type
• if $t \in T$ then $Set(t) \in T$ is collection types

Definition 2 *(attribute signature)*
Let $a \in A$ be an attribute name, $t \in T$ be an atomic type and S_p be the set of attribute signatures. An attribute signature $s_p \in S_p$, called primitive attribute signature, is defined over the set of types T as follows:
• $a : t$ is an attribute signature in S_p
• $a : Set(t)$ is collection attribute signatures in S_p
• if $s_i \in S_p, i = 1, \ldots, n$ then $a : tuple(s_1, s_2, \ldots, s_n)$ is a tuple attribute signature
• if $s_i \in S_p, i = 1, \ldots, n$ then
$a : Set(tuple(s_1, s_2, \ldots, s_n))$
is a structured collection attribute signatures

Definition 3 *(operation signature)*
Consider an operation $op \in OP$ and a type $t \in T$. Let SO be the set of operation signatures. An operation signature $s_o \in SO$ is defined by: $op(s_1, s_2, \ldots, s_n) \rightarrow t$ where $s_i \in S_p, i = 1, \ldots, n$ is a sequence of attribute signatures and $t \in T$ is the result of *op*.

Definition 4 *(type interface)*
Let I be the set of interfaces. A type interface $i \in I$, defined on $t \in T$, such that $t = \sigma(i)$, is a tuple
$< Structure, Operations >$:
$Structure$ is a sequence of attribute signatures.
$Operations$ is a sequence of signatures associated with the set of operations applicable on the type t.
The function $\sigma : I \rightarrow T$ returns the type associated with the interface.

Definition 5 *(subtype, supertype)*
Consider two types t_1 and t_2. Let $i_1, i_2 \in I$ be the interfaces of t_1 and t_2 respectively. t_1 is a subtype of t_2, noted $t_1 \leq t_2$ if and only if $Structure(t_1) \subseteq Structure(t_2)$ and $Operations(t_1) \subseteq operations(t_2)$. t_2 is a supertype of t_1. The relation $\leq$ defines a partial order on the set T.

Definition 6 *(object)*
An *object instance* or *object* $o \in O$ represents a real world entity. An object is defined by a tuple $< oid, v >$ where $oid \in OID$ is the identifier of o and $v \in V$ is a value of type $t \in T$ which represents the state of the object (attributes of o).

Definition 7 *(class)*
A class $c \in C$ represents a set of objects with the same type $t \in T$ (they have the same properties and operations). A class has an interface which is a type and an extent which is denoted by $\pi(c)$. A class is defined by a tuple $< i, extent >$ where i is the interface of t. The extent of a class $c \in C$ is the set of OID of the objects of c.

Definition 8 *(subclass, superclass)*
Consider two classes $c_i =< i_i, extent_i >$ and $c_j =< i_j, extent_j >$ in C. c_i is a subclass de c_j (noted $c_i \prec c_j$) if and only if $\sigma(i_i) \leq \sigma(i_j)$ and $\pi(c_i) \subseteq \pi(c_j)$

4.1.2 An Example using ISL

We have given above a formal description of the core concepts which provide a foundation for the definition of spatial and distributed object types of the AMUN data model.

A complete formal description of the concepts is beyond the scope of this paper. In the remainder of the paper, we will use the interface specification language (ISL) and interface query language (IQL) to present spatial object data types. ISL and IQL are closed to ODL and OQL defined by ODMG-93.

We now introduce a simple example of schema that to illustrate the ISL syntax and underly the spatial type definitions given in the remainder of the section. The example represents a schema which models a parcel (plot) of farm land. The information of interest are parcel number, the owner, the type of crop (culture), a list of values representing the harvest and eventually geographic information on the parcel.

```
create interface TParcel
  properties
    attribute integer Parcel#;
    attribute string Owner;
    attribute set(float) Harvests;
    attribute string CropType;
  operations
    void write_owner(in string name);
    string read_owner()...
```

A class description corresponding to `TParcel` is given below. The extent of the class `CParcel` named `EParcel` is a container for the oid of parcel objects.

```
create class CParcel of TParcel
 extent EParcel
```

4.2 Spatial types

The data model AMUN comprises a set of predefined spatial data types and associated operators. The predefined spatial data types are based on a subset of the spatial types of the OpenGIS specifications [8]. OpenGIS spatial type are described by well-known structures which represent their semantic. AMUN uses a subset of the OpenGIS hierarchy. `Geom` is the highest spatial type in the hierarchy. It represents general geometric information. The spatial data type `CoordinateGeometry` is a subtype of `Geom`. It models spatial object which have coordinate informations. The low level basic spatial types of the hierarchy are: `Point`, `LineString`, `Polygon`, ... The spatial types can be included in a general type to model the geometric aspect of the corresponding object. For example, an attribute `Geometry` with type `TSIMPLEPOLYGON` can be added to the type `TParcel` above to model the geographic properties of the farm land. The new definition for `TParcel` is shown below.

```
create interface TParcel
/* A parcel with its geometry */
  properties
   attribute integer Parcel#;
   attribute string Owner;
   attribute set(float) Harvests;
   attribute string CropType;
   attribute TSIMPLEPOLYGON Geometry;
  operations
   void write_owner(in string name);
   string read_owner()...
```

The spatial type `TPOLYGON` is used to model simple polygonal surface without hole and island. The chosen representation for polygons is by vertex.

```
create type interface TPOLYGON
/* Ring (simple polygon)*/
properties
 attribute set(Coordinates) Vertex;
operations
 float surface;
 boolean intersect_poly(Polygon: TPOLYGON);
 boolean intersect_line(Line: TLINE)
```

4.3 Distributed Spatial Object Concepts

This section presents concepts for defining virtual global view of classes from heterogeneous spatial data sources. We define types and concepts that can be used to: 1) restructure the value of object, thus allowing multiple representations derived or calculated from the values of an objet; and 2) allow aggregation of information from different sources. To achieve this, we define the concepts of calculated attributes, calculated extents, concrete class and virtual class. We give in the next subsection formal definitions of these concepts.

4.3.1 Calculated attributes and calculated extents

A calculated attribute is an attribute whose value is obtained by the evaluation of an expression on the objects of a class. The signature of a calculated attribute is defined as follows.

Definition 9 *(calculated attribute signature)*
Let S_c, $s_c \in S_c$ denotes the set of calculated attribute signatures and the signature of a calculated attribute respectively. The signature of $s_c \in S_c$ is defined by $< s, expr >$ where $s \in S_p$ is an attribute signature and $expr$ is a well-formed expression over the sets A and OP

For example, below is a calculated attribute to represent the best harvest from the type `TParcel` define above.

```
calculated attribute
 integer BestHarvest = Max(Harvests)
```

A calculated extent is a class extent obtained by evaluating a calculated attribute on the instances of a class.

Definition 10 *(calculated extent)*
Let $e \in E, e_i \in E, i = 1, \ldots, n$ be class extents and E be the set of all class extents. Let f be a formula or an

expression. A calculated extend is computed from one or more class extents as follows:

• $\Sigma_f(e) \in E$ is a calculated extent from the extent of a single class c of C as follows: It contains the objects of c verifying the formula f. We distinguish two cases:

- if f is a logic formula, then $\Sigma_f(e)$ contains the objects of c verifying the formula *f*

- if f is an expression which compute a new value for each element of c, then $\Sigma_f(e)$ contains the same objects as extent of C.

• $\Sigma_f(e) \in E$ is calculated from two or more extents $e_i, i = 1, \ldots, n$ of C, $\Sigma_f(e) = \bigcup_{i=1,\ldots,n}(e_i) \in E$ is a calculated extent defined by the set union of the concrete/calculated extents e_i.

4.3.2 Concrete and Virtual class

A concrete class implements a type interface on a local information system. It is used to describe objects really stored in different data repository. A concrete class is defined as follow:

Definition 11 *(concrete class)*
A concrete class is defined by $< i, extent, w >$ where $i \in I$ is an interface and extent represents the set of objects of the concrete class. Wrapper w contains an implementation for the interface i and is capable of retrieving the corresponding physical objects from a local extent.

Figure 3 shows two concrete classes associated with the type `TParcel` defined above. The classes are defined in two wrappers $w1$ and $w2$ to represent parcel information modeled by two GIS: GIS1 and GIS2. A definition of the concrete classes using the ODL language is given below.

```
create concrete class CParcel1 of TParcel
 extent EParcel1 on w1 ;
create concrete class CParcel2 of TParcel
 extent EParcel2 on w2 ;
```

A Virtual class which is used to model distributed spatial objects. The definition of virtual class is based on concrete classes, calculated attributes and calculated extents. A virtual class is defined as follows.

Definition 12 *(virtual classes)*
A virtual class vc is a class which is not implemented on any wrapper. It is defined by $< i, extent >$ where i is a type interface and extent is the set of objects of the virtual class.

In our approach, virtual classes are used, at the mediator level, to create global view over distributed information sources. A virtual class enables a user to create virtual objects by restructuring 1) the state or the value of one or more objects; or 2) the extent of one or more concrete classes. We distinguish two cases for the definition and creation of a virtual class.

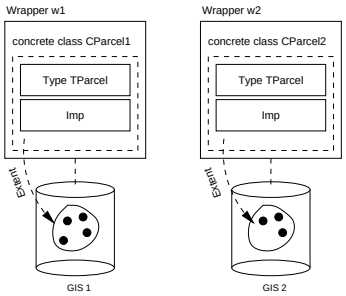


Figure 3: Concrete Classes

Case 1: restructuring attributes to create a virtual class. A virtual class is created by restructuring attributes of existing objects. The states of the objects of the virtual class are defined by applying an expression on the state of existing objects. Furthermore, the state of one or more objects can be aggregated to create virtual objects which do not exist in any component GIS. Instead, they are created or calculated on demand from objects in existing systems. For example, to construct a city map, a user may have to aggregate objects from two sources: a GIS which models road traffic information and a second GIS which contains information on buildings.

The following example (figure 4) presents the definition of a virtual class for the type `TParcel` defined above. Two calculated attributes `MeanHarvest` and `Surface` are used to define new values from the information contained in the type `TParcel`. The resulting type, which is a subtype of `TParcel`, is called `TParcelHarvest`. `CVPArvestHarvest` is the created virtual class and `EVParcelCrop` is its extent. Each object of the new virtual class `CVParcelHarvest` is calculated from an element of the concrete extent `EVParcel`.

```
create interface TParcelHarvest :: TParcel
 calculated attribute float MeanHarvest
 calculated attribute float Surface
create virtual class CVParcelHarvest
 of TParcelHarvest
 as select Parcel#: p.Parcel#, Owner: p.Owner,
  CroptType: p.CropType, Harvest: p.Harvest,
   MeanHarvest: Mean(p.Harvest),
   Surface: surface(p.Geometry)
     from p in ESelf
/* ESelf describes the extension of the class */
  extent EVParcelCrop as EParcel1
```

Case 2: restructuring extents to create a virtual class. The definition of a calculated attribute can be a logic expression which is used to select the objects contained in the extent of a virtual class. The logic expression represents an applica-

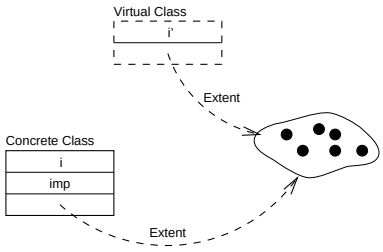


Figure 4: Virtual class with calculated attributes

tion semantics which must be verified by the elements of the new virtual class. We distinguish two cases:

• The semantic associated with the calculated attribute is applied on the elements of a given class (concrete or virtual). In this case, the virtual class corresponds to a selection operation which restricts the extent of the original class (figure 5(a)).

The following example presents the definition of a virtual class `CVParcelWheat1` for the type `TParcel` defined above. The extent `EVParcelWheat` of the virtual class is defined by the select query which is used to set the value of attribute `CropType` to `'wheat'`. `EVParcelWheat` is calculated from the concrete extent `EVParcel`.

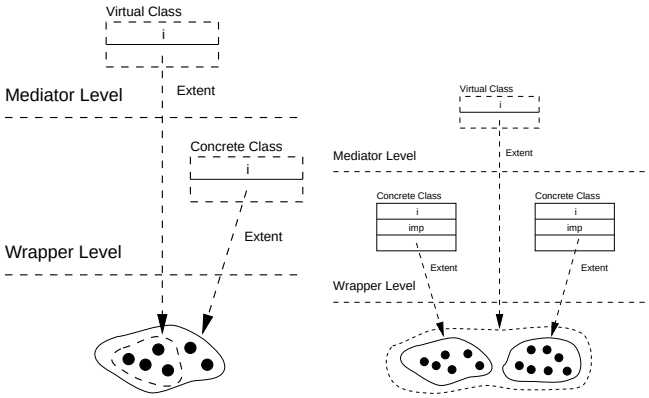
```
create virtual class CVParcelWheat1 of TParcel
 extent EVParcelWheat as
  select p from p in EVParcel1
   where p.CropType='wheat'
create virtual class CVAllParcel of TParcel
 extent EVAllParcels as union(EVParcel, EVParcel)
```

• The semantic associated with calculated attributes is applied on the elements of two classes (figure 5(b)). All objects which verify this semantic (whether or not from the same class) are grouped to form the extent of the virtual class. Note that the original objects may have different structural properties. The virtual class interface, which must be compatible with the structural description of the objects, will be used to access and manipulate the resulting virtual objects.

The following example presents the definition of a virtual class `CVAllParcel` as a union of the two concrete classes `EVParcel1` and `EVParcel2` (figure 5(b)). The new class represent all parcels modeled by GIS1 and GIS2. The resulting virtual extent is `EVAllParcels`.

5 An operational architecture based on CORBA

Figure 6 shows the operational architecture of ISIS. It is based on OMG distributed object management model and its core component CORBA. The functional architecture of ISIS is implemented on top of CORBA. The common data model (AMUN) and a compliant OQL query language are



(a) Extent calculated by formula

(b) Extent defined by union operation

Figure 5: Virtual classes and calculated extents

used to realize the components (mediators and wrappers) of the functional architecture. Our architecture can be used to incorporate different GIS such as GIS Arc/Info, Illustra, GRASS into a multi spatial information system.

To achieve GIS interoperability, we decompose the architecture in different layers [13, 1]. The most important layers are the physical data store layer, the GIS functionalities layer, and the data model layer. Each layer provides a set of schema and services to implement the functionalities of the next higher layer. Platform (hardware and network layers) interoperability is handled by CORBA functions and services. CORBA's Object Request Broker (ORB) supports interconnectivity for high level services. These services deal with high level expressions over OQL and AMUN in an homogeneous environment. Thus, the different services can reside on different machines of a network, and can be implemented in different languages (e.g. ADA, C++, Java, ...).

In our approach, users applications are clients for the mediators. When a query formulated in a global language is submitted to a mediator, the global query processor in the mediator determines the data and the operations requested by the query. The mediator sends multiple sub-queries to the wrappers taking part in the execution of the global user query. Each wrapper receives only sub-query that can be processed locally, that is the required and operations are available on the local site. The wrapper uses transformation rules to translate the query to the local GIS language and provides secure connections to the requested data. It submits the query to the local GIS, converts instances of the result, and send the result to the mediator. Then results from participant sites are combined into a final result in the mediator. The data are mapped to local format and presented to the user.

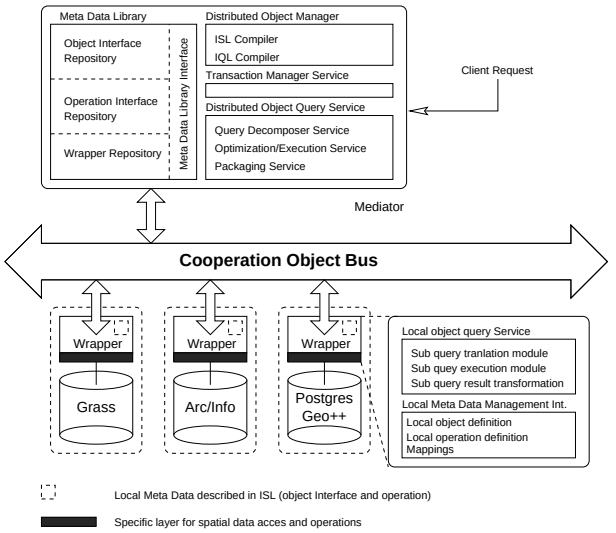


Figure 6: Operational Architecture

6 Conclusion

We describe ISIS project that is specially design to address issues relevant to the design of distributed interoperable GIS.

We introduce a distributed spatial object model AMUN as an extension of ODMG-93. The main contributions in this paper are the following:

• concepts to deal with distribution and heterogneties: calculated attributes, calculated extents, concrete and virtual classes;

• spatial data types to deal with spatial objects;

• functional extensions to deal with distributed spatial operators. While distribution is addressed with the materialization level, the first two levels (semantic and concrete level) are used to resolve heterogeneities by grouping spatial operation with the same semantic.

We also present the architecture allows sharing of both data and operations. It's major components are: 1) wrappers (one per local GIS) for resolving heterogeneity in the cooperative environment, 2) mediators for coordinating cooperative tasks such as services communications or query dispatching. The main advantage of the architecture is it brings core software components to be used in different contexts and thus it allows flexible and extensible cooperation in different environments such as WEB [1], federated GIS.

The initial phase of our project is devoted to the definition of the data model. Our future work will focus on spatial query processing to handle the distribution and sharing not only of spatial objects but also of specialized spatial operators.

References

[1] Y. Bishr, M. M. Radwan and J.Pandya, SEMWEB - A Prototype for Seamless Sharing of Geoinformation on The World Wide Web in a Client/Server Architecture *Proc. of the Third Joint European Conf. & Exhibition on Geographical Information*, pp.145-154, 1997.

[2] The Common Object Request Broker : Architecture and Specifications, *Object Management Group*, OMG Document Number 91.12.1, December 1991.

[3] M. Coyle, S. Shekhar, D. Liu, S. Sarkar, Experiences with Object Data Models in Geographic Informations Systems, Technical Report, TR 95-10, 1995.

[4] B. David and A. Voisard, A Unified Approach to Geographic Data Modeling, Technical Report 9316, *University of Munich (LMU)*, September 1993.

[5] K. Gardels, Open GIS and On-Line Environmental Libraries, *ACM SIGMOD Record*, 26(1), March 1997.

[6] E. Leclercq, D. Benslimane, K. Yétongnon, A Distributed Object Oriented Model for Heterogeneous Spatial Databases, *Tenth Int. Conf. on Parallel and Distributed Computing Systems*, 1997.

[7] S. Nittel, J. Yang, R. Muntz, Mapping A Common Geoscientific Object Model to Heterogeneous Spatial Data Repositories, *Fourth ACM Workshop on Advances in Geographic Information Systems*, November 1996.

[8] OGIS Consortium, The OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing, Revision 1, *OpenGIS Consortium*, TC Document 96-015R1, 1996.

[9] E. Pitoura, O. Bukhres, and A. Elmagarmid, Object Orientation in multidatabase systems, *ACM Computing Surveys*, 27(2), pp.141-195, June 1995.

[10] M. Scholl and A. Voisard. Object-Oriented Database Systems for Geographic Applications: An Experiment with O2, *in The O2Book*, F. Bancilhon, C. Delobel and P. Kanellakis (Eds.), Morgan Kaufmann, San Mateo, California, 1992.

[11] A. Sheth and J. Larson, Federated database systems, *ACM Computing Surveys*, 22(3), pp. 183-236, September 1990.

[12] N. Tryfona and J. Sharma, On Information Modeling to Support Interoperable Spatial Databases, *Proc. of the 8th Int. Conf. on Advances Information Systems Engineering, CAiSE'96*, Vol. 1080, pp. 210-221, Springer, 1996.

[13] A. Voisard and H. Schweppe, A Multilayer Approach to the Open GIS Design Problem, *Proc. of the 2nd ACM GIS Workshop*, pp. 23-29, 1994.

[14] M. Worboys, H. Hearnshaw, D. Maguire, Object-oriented data modelling for spatial databases, *Int. J. of Geographical Information Systems*, Vol. 4, Num. 4, pp. 396-383, 1990.

[15] M. Worboys and S.M. Deen, Semantic Heterogeneity in Distributed Geographic Database, *SIGMOD Record*, Vol. 20, Num. 4, pp. 30-34 , 1991.