

Supporting Interoperation of GIS Objects¹

Silvia Nittel, Richard R. Muntz
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
{silvia, muntz}@cs.ucla.edu

Extended Abstract

Today, a promising approach to large-scale interoperability in geographic information systems (GIS) is a GIS implemented via a distributed object management system (DOMS) (e.g. CORBA, Java, OLE/COM) based on a standardized spatial object model (e.g. OGIS [1]). Geographic entities are available as location- and platform-independent objects, and are accessible via their interface, i.e. their attributes and a set of operations. Using a DOMS, much of the hardware, network and communication heterogeneity between interoperating GIS objects on different platforms and machines is handled automatically by the DOMS software, and is of no concern to the user. Also, via the notion of strongly typed object interfaces, type checking is also automatically performed at compile time to ensure that correct argument types are passed between interoperating GIS objects.

However, to resolve the correctness of an argument passed from a GIS client to a GIS object, it is not sufficient to only verify the correctness of the *type* of the argument, but it is necessary to ensure that the *content* of an argument is matched. By content, we mean constraints related to the actual properties of an argument that are not easily captured in the type hierarchy. For example, let us assume that the operation “intersect()” of the GIS object type “LineString” has the following signature:

```
Geometry intersect(in Geometry other);
```

“Intersect” expects an input argument of type “Geometry”. To perform the intersection operation correctly, both geometries have to be in the same spatial reference system (SRS), i.e. in the same coordinate system with the same metrics for the axes. However, the requirement of this argument conformance rule is not explicit in the operation’s signature, and can not be automatically checked at compile time. Also, the client cannot be sure in which spatial reference system the returned “Geometry” object will be defined, and cannot specify explicit rules for the kind of spatial reference system it would require to be returned (e.g. for displaying objects). Other examples include a GIS service that handles the type “Image”, and has to apply operations to images of different formats, resolution, size, compression, etc..

Today’s DOMS system offer little help to resolve issues of interoperability of GIS objects regarding the content since its interpretation is usually based on property values of the user-defined GIS object rather than supported through a type system. This makes it difficult to define general conformance rules between interfaces in distributed GIS systems. Today, it is the server’s task to check which input parameters can be handled, and which cannot, and either provide implementation for different kinds of parameter content or return an error message. On the other hand, a client might have to explicitly conform to “content” rules of a service (e.g. providing a geometry object within a specific SRS), and have to re-

¹ Published in 1st Int. Conference Interoperating Geographic Information Systems (Interop'97), Santa Barbara, California, December, 1997.

match the return value from a GIS server to its requirements. This is error-prone, cumbersome and makes server and client code bulky.

In the proposed paper, we examine the issue of GIS object property typing with a view to formalize how to cope in a systematic way with the heterogeneity of GIS object properties in a general distributed environment. Key issues for which we discuss solutions concern the problem of how to decide whether two given GIS objects are compatible based on the *content* of their arguments, or can be made interoperable through the introduction of ancillary mechanisms such as *object property reconciliators*.

In our approach, we introduce a type model in which we interpret a type specification as a GIS object property descriptor. A GIS object property descriptor consists of a generic type descriptor such as SRS, IMAGE, or GRID, and a set of attributes specifying the properties of a specific GIS object; for example, the notion IMAGE[640,480,JPEG] can be used to denote the image properties of a map object with width, height and encoding equal to 640, 480 and JPEG encoding, respectively. This type model supports the definition of a variety of GIS object property type relationships.

In particular, we define subtyping rules supporting the conformance of GIS object interfaces regarding the content of arguments; e.g. SRS[Cart,2D,1,km] describes a 2-dimensional spatial reference system with a Cartesian coordinate system, tickmarks as multiples of 1, and kilometer as unit. A SRS property of this value can *substitute* a SRS with the value SRS[Cart,2D,1,m]. In some cases, a subtyping relationship might be too restrictive, i.e. for interoperability it is sufficient if two GIS types support at least one common kind of property. If they support more than one common kind of property, a GIS object property negotiation may take place to choose the property that is actually used.

If two GIS object properties do not support a common property requirement, they might still be interoperable, if it is possible to use a property reconciliator such as a converter of a spatial reference system or a converter for image encoding. We also use the GIS object property type model to define the characteristics of a facility supporting a dynamic reconciliator based on GIS object property type relationship.

Using this GIS object property type model provides for automatic reconciliation of the *content* of arguments passed between two GIS objects. The mechanism is reusable for all kinds of different GIS objects. Relinquishing both services and clients from the task of checking the arguments makes their implementation simpler and more lightweight. Since the GIS object property reconciliation mechanism is extensible, i.e. new description types and conformance rules can be added, GIS services can profit from the extension without any code having to be added to their implementation.

References:

- [1] Buehler, K., McKee, L.: Introduction to Interoperable Geoprocessing, OGIS Project Technical Committee of the Open GIS Consortium, Inc., OGIS TC Document 96-001, 1996.