CORBA–Based Interoperable Geographic Information Systems

H. –Arno Jacobsen[*]   Agnès Voisard[†]

TR-98-011

April 1998

Abstract

A new generation of geographic information systems (GIS) emphasizing an open architecture, interoperability, and extensibility in their design has received a great deal of attention in research and industry over the past few years. The key idea behind these systems is to move away from the traditional monolithic view in system engineering, to an open design embracing many co–existing distributed (sub)–systems, such as database management systems (DBMS), statistic packages, computational geometry libraries and even traditional GIS. While some success has been achieved in the area of geospatial data integration (data models and formats), it is still unclear what common services these open GIS should provide and how their design would benefit from available distributed computing infrastructures. This latter question is especially interesting with regard to the increasing attention that object–oriented distributed computing infrastructures have received over the past few years in the community.

In this paper, we describe a generic open GIS with an emphasis on the services it should provide. We then study the design of such a system based on object services and features provided by the Common Object Request Broker Architecture (CORBA). We also report on the use of the CORBA platform for implementing a fully–operational distributed open GIS. We conclude by arguing for a closer integration of GIS functionality into the CORBA architecture, as already done for the medical and financial domains.

[*]Author's permanent address: Institute of Information Systems, Humboldt University, Spandauer Str. 1, 10178 Berlin, Germany. E-Mail: `jacobsen@wiwi.hu-berlin.de`

[†]Author's permanent address: Institute of Computer Science, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, Germany. E-mail: `voisard@inf.fu-berlin.de`

1 Introduction

A GIS enables to input, store, query, analyze, display and select spatial (geometric, topological) data as well as non-spatial data [MGR91, LT92, Güt94]. Traditionally, such functionalities are encompassed within the same system. Basic design principles of open GIS move away from this integrated view. There are many reasons motivating this design. Following are some of them. First of all, as far as the data itself is concerned, geographic data sources are distributed by nature and highly heterogeneous. Take for example applications based on sensor data (e.g., seismic applications), GPS data, or data from public agencies, such as environmental data. To integrate, analyze and work with the data in one seemingly local system, it needs to be downloaded across the network and appropriately represented locally.

Next, as far as (geo)data processing is concerned, there already exist specific systems that are specialized in performing certain tasks efficiently. Take for example a shortest path algorithm in a routing application. Why should such an algorithm be coded within a traditional GIS? It is clearly more efficient to call a (stand alone) expert system or a powerful geometric algorithm from a library to compute the result. In a traditional environment, the integration of such an algorithm requires a great deal of effort (adaptation to the system, coding of the algorithm, etc.). If the link with a geometric library were made, using (e.g., integrating and calling) such an algorithm would be trivial. In the sequel, we refer to such "tasks" as operations or *services* provided by *participating systems*. These systems are potentially distributed over a network.

Another motivation is driven by the increasing use of computer networks during the past decade. With the emergence of distributed system technology, it becomes much easier to consider a theoretically unlimited number of services available from participating systems, distributed across the network.

As a consequence, open GIS allow to handle applications that could not be handled by a traditional GIS, or that were at the edge of being handled by it. Spatial decision making applications, business mapping and geomarketing applications, which require intelligent modules to correlate and analyze data from different sources (e.g., for simulation or for selling strategies) are examples of such applications.

One might wonder about the difference between open GIS and open systems in general. Geographic applications require particular attention because of: (i) the existence of large amounts of data; (ii) the existence of complex and highly-structured data; (iii) the co-existence of many different geographic formats; (iv) the increasing trend towards reuse of geographic data all over the world; (v) the fact that distributed data and operations are equally important; (vi) the existence of specific requirements due to the nature of the data, such as the integration of a specific set of functionalities, data visualization and modeling.

In this paper we focus on the design of a generic open GIS based on the features and services provided by a standardized distributed computing infrastructure. This involves considering the following aspects:

- Cooperation of heterogeneous components in one global system,
- interoperability between separately designed sub–systems,
- integration of components distributed across the network,

- dynamic system extension, (i.e., the possibility of plugging components into application at run–time).

The rest of the paper is organized as follows. Section 2 gives an overview of related issues and previous work. Section 3 presents a requirement analysis of open GIS. We introduce a reference example borrowed from the domain of geomarketing, which serves to illustrate the general principles discussed in the remainder of the paper. Section 4 provides a brief overview of CORBA and shows, first abstractly, how open GIS can be based on CORBA technology. We then take up the example presented earlier and motivate how the identified requirements are fulfilled given CORBA features and services. More generally, this section demonstrates how component architectures profit from a unifying object model and a common communication infrastructure.

We conclude by arguing that open GIS architectures will greatly benefit from the unifying model CORBA presents. We raise the question whether a closer integration of open GIS ideas into the OMG's Object Management Architecture would not further contribute to the developments in both areas.

2 Related Issues and Previous Work

One task in the realization of open GIS is to consider seamless integration of data. In the database area, lots of work has been done on the access to heterogeneous repositories, also called data sources in the database jargon. The architectures of such distributed systems (e.g., [RAH$^+$96], [AAD$^+$93], [PAGM96], to list a few) is usually centralized around a mediator which communicates with wrappers, i.e., interface to data sources. Object management in integrated distributed systems is discussed in [MHG$^+$92]. Open GIS design goes beyond data source integration, but it can generally benefit from these approaches.

Software modules considered in an open GIS include statistics packages, expert systems but also specific algorithms such as locate/allocate (typical GIS algorithms used intensively in geo-marketing for instance), shortest path, which can be collected inn a library. Such a software package or library can be integrated and called across the Internet. Recent work on the definition of a framework to call software packages remotely, e.g., [GMS$^+$97], illustrates the power of this approach. However, using this framework requires some adaptation effort for open GIS design as the crucial notion of state is missing. In other words, a mechanism such as a (database) transaction has to be incorporated in it to allow multiple access to the data and a control on the overall system.

Regarding geodata exchange and standardization, it is worth acknowledging the work that has been done in the past few years by standardization bodies. Among them are ISO/TC211 in the US, CEN/TC287 in Europe, SAIF in Canada, and Interlis in Switzerland. Significant work has been achieved by the Open GIS Consortium for the past two years. The Open Geodata Interoperability Specification (OGIS) [OGI96b] provides a framework to create software that enables users to access and process geographic data from a variety of sources across a generic computing interface. The approach is geared towards implementations on top of various distributed computing platforms (DCP), and is articulated around the Open Geodata Model [OGI96a] and the OGIS Services Model. It does not claim to be a standard but given the success of this enterprise it might become a *de facto* standard. However, to our knowledge, the mapping onto DCP, which is one of our goals here, has not been studied so far in this context.

[VS98] is concerned with conceptual design issues and proposes a layered decomposition of an open GIS architecture. A high level of abstraction, the application level, is refined until the system level is reached. In-between are (i) the abstract services level, at which systems is a set of capabilities, and (ii) the concrete services level, where distribution over the participating systems is managed. At each level, the system is described using the same 3-feature model: *data*, *operations* one can perform on this data and *sessions*, the graph of operations invoked to answer a given query, also defined at each level. Switching from one level to the other one in the hierarchy is achieved through the use of metavariables, both on the systems and on the data itself. In the sequel, we will make reference to this framework.

One of the goals of our work is to demonstrate how CORBA [Obj95b] can be employed in the implementation of open GIS. CORBA has been widely recognized as a standard for achieving system interoperability and as base technology for distributed computing environments. Based on a rich object model, it is distinct from other approaches such as OSF's DCE [OSF96] or Microsoft's OLE/COM [Bro96]. OLE/COM [Bro96], which is purely object–based[1], operates, so far, only on the company's proper operating systems thus not permitting interoperability across different platforms and operating systems, a major drawback for open GIS. Bridging technology for interfacing CORBA to OLE/COM has recently been initiated by the OMG [Obj96].

Other teams have exploited the benefits of CORBA for distributed system design. Much work has been done on integrating different data sources across the network (see for instance [MMS$^+$96, Ama97]). Our approach is different as we focus on servicesn rather than on data integration. [BR94] describes a system for air quality control, featuring heterogeneous components, distributed systems and integration of legacy code. However, it uses PVM (partial virtual machine) to achieve distribution and does not address issues related to the use of CORBA.

Closer to our work is the architecture described in [KKN$^+$96], which presents an environmental systems incorporating the GIS GRASS, the World Wide Web and CORBA. However, the focus in mainly on the actual implementation and there is no highlight on the use of CORBA services for such new GIS generation.

3 Open Geographic Information Systems

This section is devoted to the main characteristics of an open GIS. It starts with a requirement analysis of open GIS. We then describe the architecture of a generic open GIS and discuss its main modules. We refer to such a generic system as an "overall system" composed of participating systems, or components, or simply systems for short.

As far as the participating systems are concerned, we make the distinction between repositories (data servers) and software packages (computational entities). Note that this difference is not exclusive and systems like GIS or DBMS belong to both categories. An open GIS aims at incorporating them within a predefined framework while keeping their

[1]A detailed comparison of both would lead too far. It is, however, worth noting that OLE/COM does not support inheritance which is a key object-oriented engineering concept.

ontology. Each participating system specifies its interface in a common interface definition language, as for instance a wrapper around database management systems (wrappers help access heterogeneous data sources, by transforming access and update requests to interface expressions, e.g., SQL, URLs).

3.1 Requirement analysis for open GIS

Open GIS aims at providing a single coherent framework that allows to integrate the different physically distributed and heterogeneous compnonents into one overall system. The overall system should provide one consistent interface shielding the user from the various components interacting and cooperating in the system.

This goal in mind and given the observations listed in the introduction, we can characterize the requirements of open GIS as outlined below. It is important to note that most of these requirements will also apply to distributed system in general. GIS specifics are summarized thereafter. The following list enumerates and motivates the individual requirements we attribute to open GIS.

1. Component integration.

   The overall system must be designed such that it facilitates the integration of different components. This is further complicated by the fact that most components will have been developed separately without integration as ulterior design goal. This, for example, is the case for legacy systems, often not even developed inhouse. Furthermore, component integration suffers from the fact that much prior developed code will be available in different programming languages, also developed with different methodologies in mind.

2. Dynamic extensibility.

   The system should allow the discovery of new components at run–time. It should be possible to integrate and use these components without recompiling or relinking the entire system. This is important to exchange components for maintenance purposes, as well as extend the overall system by adding new functionality.

3. Self-describing components.

   In order to allow dynamic system extension, the system and/or each component must provide means to exhibit its interface and ideally its functionalities to other participants.

4. Software engineering requirements.

   From a software engineering point of view portability of the design is a key requirement, since the framework should be easily adaptable to slightly different applications.

5. Scalability, fault–tolerance and performance.

These are desirable requirements which do not play a primary role for the application domain we are targeting. For more mission-critical applications these requirements will, however, dominate.

The system should scale gracefully when adding new hardware or software components. Moreover, the system design should foresee automatic resynchronization mechanisms after failures or crashes of individual components.

Impetus of geodata peculiarities on system design

The requirements outlined above are fundamental to the design of an open and extensible distributed system in general. The nature of geographic data, however, adds another set of characteristics which cannot be neglected in the more specific GIS design. These are:

- The need to handle huge volumes of data (MBytes for map, GBytes of atmospheric data). This prohibits the use of classical techniques (e.g., for data exchange and storage).
- The co–existence of many proprietary systems with their own formats for data representation. This renders excessive data transformations necessary and calls for the incorporation of data semantics.
- Support for *complex* data models due to highly structured data and to specific data–bound operations. These include the above-mentioned format transforms, but also typical geographic functions such as transformations, generalization and aggregation. The fact that in open GIS data is gathered from heterogeneous sources makes it difficult to handle these functions which are already tricky in a monolithic GIS.
- The growing interest in, and need to, reuse data around the world, which puts constraints on data collection, representation and storage techniques.
- Extensive user interaction in handling the different data sources, due to the visual and hyper aspects of geospatial data.

In Section 4.2 we demonstrate how CORBA features and services may be used to meet these requirements.

3.2 Architecture of a generic open GIS

Prior to presenting the architecture itself, let us describe an illustration example. A whole family of applications which were at the border of being GIS applications, as for instance collaborative spatial decision making or geomarketing applications, will greatly benefit from the integrated approach of the new GIS generation. The typical simple pattern in this type of applications is (i) to visualize a geodata set (usually regional maps[2] we use here maybe together with addresses), then (ii) overlay sectorial information, and then (iii) answer a precise question or check hypothesis in order to draw conclusions. The following simple example borrowed from this area will be a reference throughout our study:

[2]From now on we use the term 'map' to refer to the object stored in a repository, and not to a frozen representation of it (with legend) displayed on the screen, for example.

An example open GIS application

Suppose that a supermarket chain wants to find out the best location to open a new store, based on (i) the map of a neighborhood (e.g. downtown Berkeley), and (ii) the distribution of households whose average expense per visit in a supermarket is more than $ 200.

The steps necessary to handle this study are:

1. Get a map of downtown Berkeley.
   We suppose that the map is stored in a database called DB1.

2. Get the information (addresses) regarding households in this area whose average expense per visit in a supermarket of this chain is more than $ 200.
   We suppose that the information is stored in a file system called FS1 (whose owner is the supermarket chain).

3. Overlay the two maps and display the result on the screen.

4. Draw the conclusions manually or call a spatial allocation module which computes the best spot.
   A spatial allocation module is available in GIS2.

Basic functionalities of open GIS

Figure 1 shows one possible architecture of an open GIS. Note that the system is centralized around an integrator and that other architectures, with a more direct communication among the modules, could have been chosen. For a discussion on this issue see [AKC94]. For a sake of simplicity, only the main modules are depicted. First, the user inputs queries/commands into the global system via a graphical user interface. These queries and commands are a graph of instructions that will be performed by one or many subsystems. The integrator module will dispatch the instructions to the subsystems. At this point a remark on transparent distribution is noteworthy. Previous work on open GIS integration is based on transparency, that is, operations are called without exact knowledge of the systems that provide them. A mechanism based on global knowledge on the overall system and on heuristics (as the ones proposed in [WWC92] or in [VS98] at the concrete services levels) is in charge of finding the appropriate capabilities over the network. It is clear that such a mechanism could be bypassed if the user needs to run a specific algorithm from a specific system.

Brief description of the modules

The *graphical user interface* (GUI) provides access to the overall system in order to display, query and address commands to the open GIS and to systems in particular (i.e., to their data and metadata, see next subsection). It can be provided by the GUI component in the CORBA facilities which could be based for instance on ArcView [Dan94]. or on a graphical library. In any case it interacts directly with the *query manager* which is in turn implemented using tools such as Lex (lexical analyzer)/Yacc (parser). Figure 2 shows a screen

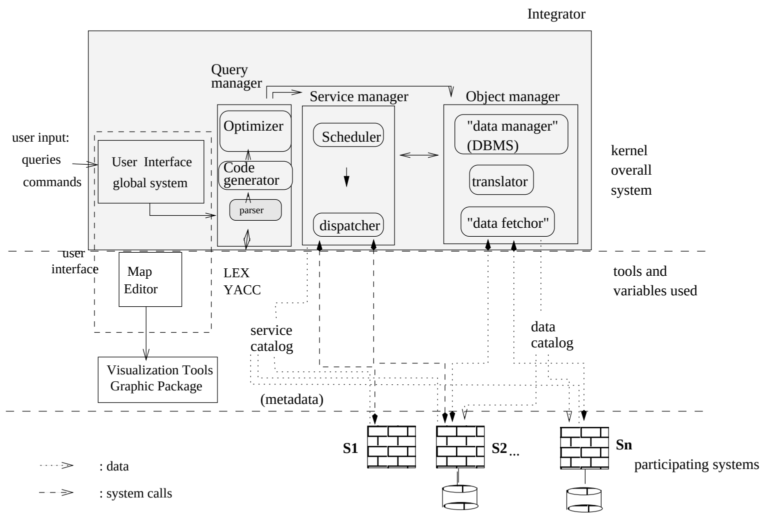


Figure 1: An architectural view of a generic open GIS.

hardcopy of our GUI which is currently under development [Mei97]. Maps are displayed in windows. Queries are entered in the query editor using a spatial query language which is not our focus here, and the generated code can be visualized in a text editor. All available services and data, together with basic operations on them, can be visualized via popup menus.

The *integrator* (somewhat similar to the notion of mediator of databases) is composed of a *query manager*, an *object manager* (OM) and a *service manager* (SM).

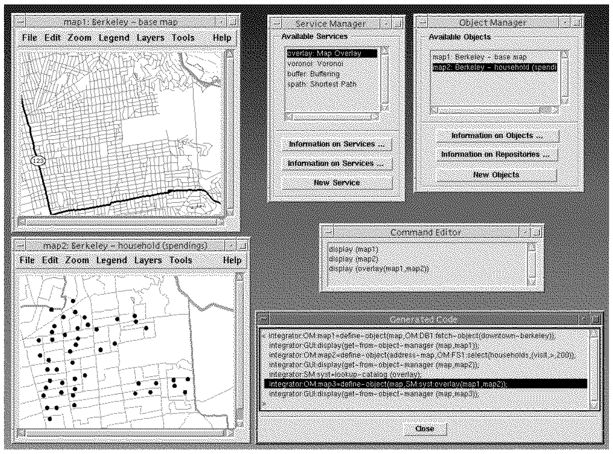
The query manager is composed of a code generator and an optimizer. It dispatches the appropriate queries among the object manager and the the service manager.

The object manager is in charge of fetching relevant objects from repositories and of defining and storing them locally (given the size of the objects involved, it could use a virtual database which consists on a view on many repositories. This is beyond our study. For more information see [VS98]). Finding objects is concerned with simple selections and distributed spatial data retrieval (e.g., [Karacapadilis ACMGIS'95). The mechanism is based on the one of a distributed spatial query language.

The service manager is in charge of communicating with all services (operations) provided from participating systems such as finding them, if needed, or calling them. It is mainly composed of a *scheduler* and a *dispatcher*. It uses the service catalog, which is a dictionary of the resources together with information about the systems that provide them (many systems can provide the same services). The dispatcher invokes system calls.

Data and metadata

In the overall system, both alphanumeric data and geospatial data (or geo-data) coexist. Metadata on both kinds of data play a crucial role as they facilitate search, browsing, and


map1: Berkeley - base map
File Edit Zoom Legend Layers Tools Help
map2: Berkeley - household (spendings)
File Edit Zoom Legend Layers Tools Help
Service Manager
Available Services
overlay: Map Overlay
voronoi: Voronoi
buffer: Buffering
spath: Shortest Path
Information on Services ...
Information on Services ...
New Service
Object Manager
Available Objects
map1: Berkeley - base map
map2: Berkeley - household (spendi
Information on Objects ...
Information on Repositories ...
New Objects
Command Editor
Generated Code
Close

```
display (map1)
display (map2)
display (overlay(map1,map2))
```

```
< integrator:OM:map1=define-object(map,OM:DB1:fetch-object(downtown-berkeley));
  integrator:GUI:display(get-from-object-manager (map,map1));
  integrator:OM:map2=define-object(address-map,OM:FS1:select(households,(visit,>,200));
  integrator:GUI:display(get-from-object-manager (map,map2));
  integrator:SM:syst=lookup-catalog (overlay);
  integrator:OM:map3=define-object(map,SM:syst:overlay(map1,map2));
  integrator:GUI:display(get-from-object-manager (map,map3));
>
```

Figure 2: A screenshot of an open GIS GUI.

data integration. They contain a description of the contents of geospatial and alphanumeric data collections and are stored in a *catalog*. This includes their type, location, access rights, owner or creation date. They are used for instance to find out which resources are present in the underlying datasets, and to which form. Typical examples of queries are "what are the types of soil classification in the repositories?" or "on which socio-professional categories can I base my study?".

Services

As far as the systems are concerned, data related to the services (operations) which are provided, together with their location, also have to be stored. The *catalog* stores all this information. More precisely the catalog stores a set of services associated with a system together with a description of the system. This description includes a way to call a system and to invoke a particular service. This information is exported by the participating systems during a check-in procedure.

There is a notion of high-level services available in all geographic applications and provided by the integrator. The most trivial services include (note that in the list below they apply both to data and metadata):

- GUI: `display` (*data*), `get-from-object-manager` (*type, data*)
- Service manager: `lookup-catalog` (*service*), `lookup-dictionary` (*system,callsyntax*), `call` (*system*), `invoke` *(operation,arguments)*
- Object manager: `fetch` (*data*), `download` (*data*), `define-object` (*type,data*), `select` (*dataset,criteria*)

At implementation time, this implies the existence of low-level services such as naming, referencing etc., which are the services we concentrate on in the next section. Below is the pseudo-code corresponding to the steps necessary to run the reference example, and which is generated internally after an enduser query (using an intuitive functional language); In the sequel, for a sake of simplicity, operations are pre-fixed by the name of the system that executes them. This list of instructions corresponds to a session of the concrete services level of [VS98]. What is exactly asked to systems (as for instance in `DB1:fetch-object(downtown-berkeley)` where system DB1 is supposed to give back the map of downtown Berkeley), is not our main concern here.

As said above, objects to be considered in the overall system are handled by the object manager and visualized by the GUI if needed. In the sequel, we suppose the existence of a geospatial model which encompasses basic types (e.g., string, integer), constructors (e.g., set, list, bag), and (named) complex types which are a combination of both. Type `map` represents such a complex type and `map1`, ... `mapn` are objects whose structure is of that type. The definition of such a geospatial conceptual model is not our focus here. A model such as the one of [OGI96b] (see Section 3.3) belongs to this category.

```
< integrator:OM:map1 =
    define-object(map,OM:DB1:fetch-object(downtown-berkeley));
  integrator:OM:map2 =
    define-object(address-map,OM:FS1:select(households,(visit,>,200));
  integrator:SM:syst =
    lookup-catalog (overlay);
  integrator:OM:map3 =
    define-object(map,SM:syst:overlay(map1,map2));
  integrator:GUI:display(get-from-object-manager (map,map3));
  integrator:SM:lookup-dictionary(GIS2,spatial-allocation);
  integrator:OM:area =
    define-object(region,SM:GIS2:spatial-allocation(best-spot,map1,map2);
  integrator:GUI:display (area) >
```

3.3 The OGIS specifications

Created in 1994, the Open GIS Consortium brings together geo-data suppliers, researchers, vendors and users in order to define specifications for interoperable geoprocessing. The consortium considers new technical and commercial approaches to interoperable and distributed geoprocessing. The main contribution of this initiative is to define Open Geodata Interoperability Specifications (OGIS) [OGI96b], which is a software framework for distributed access to geodata and geoprocessing resources. It is is mainly composed of the following components:

- The Open Geodata Model (OGM) defines a representation for modeling the earth and certain phenomena in a mathematical and conceptual way. This model provides a logical view of geographic information in general, independent of any underlying data model or format. It refers to map themes, features, objects and descriptions. A geodata object is made up of (i) a spatial component made of geometries (e.g., points, lines, polygons) as well as spatio temporal referencing (e.g., projection used, coordinate system); (ii) a set of semantics components that use a catalog or a data dictionary; (iii) metadata that describes all information necessary to interpret an object within an information community (context). As it is common in geographic applications, OGIS makes the distinction between the entity-oriented model, which leads to features, and the field-oriented model, which leads to coverages.

- The OGIS Services Model is defined for implementing services for geodata access, management, manipulation, representation and sharing between Information Communities. This model is made of "component services" (modules) used by developers in any geospatial applications. In the current state of the OGIS specification they include (we do not explain them in detail as their name is self-explanatory. For more information see [OGI96b], Chapter 7): Feature catalog, Reference system catalog registry, Semantic transformation, Operation registry, Types registry, Traders (to locate items of interest) and Query service (which is modeled after the OMG Query Service). All these modules are defined with respect to the object model. It is clear that the querying service, hub of the overall framework, will require a major effort in the future in order to provide all necessary functionalities.

This section has shown the main features that are handled in a generic open GIS. We use the term generic as we cannot get into the detail requirements from various information communities. However, we showed the existence of a constant set of functionalities. Next section concentrates on CORBA and on how it can be used to design such open geographic information systems.

4 CORBA–Based Open GIS Design

As outlined above, distributed information systems in general, and geographic information systems in particular, are composed of heterogeneous components, distributed physically, with the need to interoperate as one overall system. Our main objective is to show how open geographic information systems may benefit from distributed computing infrastructures such as OMA [Obj95b], OLE/COM [Bro96], Tina–C [TIN93] or DCE [OSF96]. In this section we show at what level the features and services of these infrastructures intervene in the overall GIS design.

We have chosen the Object Management Architecture (OMA) as underlying framework to illustrate the key design principles. The OMA is an open specification for object-oriented distributed computing. Its communication core, the Common Object Request Broker Architecture (CORBA) specifies the actual communication infrastructure.

CORBA addresses directly several of the requirements of an open GIS, as we will demonstrate below. Furthermore, bridging technology exists to interoperate CORBA–based systems with other technologies, such as OLE/COM and DCE, so that we are not imposing a restriction with our choice of infrastructure.

In Section 4.1 we give a brief overview of the key concepts of the OMA architecture, emphasizing on the object request broker. In Section 4.2 we take up the requirements of open GIS systems discussed above and show how to realize them with CORBA technology. In Section 4.3 we propose a closer integration of open GIS into the OMA architecture by factoring out domain specific needs of geographic information systems, which leads to a separate component, namely a *CORBA–GIS–facility*.

4.1 The Common Object Request Broker Architecture

The Common Object Request Broker Architecture specification, CORBA 2.0, [Obj95b] is an emerging standard for heterogeneous distributed object computing proposed by the Object Management Group (OMG). CORBA is part of a larger architectural framework – the Object Management Architecture (OMA) – for distributed object computing at large. The OMA specification [Obj90] was initially defined by the OMG in 1990. It specifies four interface categories centered around the Object Request Broker (ORB). These categories are:

1. The *Common Object Services* which provide basic services to all components of the architecture.
2. The *CORBA Facilities* which provide support for different application domains.
3. The *Domain applications* which are the actual domain specific user written components.
4. The *Object Request Broker* (ORB) which constitutes the actual communication infrastructure.

The *Object Request Broker* (ORB) defines interfaces and operations for passing and receiving requests between local or remote objects. The ORB is commonly referred to as the object bus, or simply bus. The functionality implemented by the ORB is supported by the *CORBA Common Services* which provide basic low–level functions to application objects and other objects located on the bus. These services include Naming service, Event service and Trader service among others [Obj95c].

The *CORBA Facilities* are intended as higher–level application services. The only currently standardized facility is the *Distributed Document Component Facility* based on prior work by CI–Lab's OpenDoc standard [CI-96]. Other parts of the CORBA facilities are in the RFP–phase[3] of the standardization process, such as standards for mobile coding, data interchange, business object frameworks and internationalization [Obj95a]. Committees for standardizing domain-specific frameworks are currently working with domain experts in medicine, finance, and telecommunication, among others.

CORBA provides a uniform object–oriented communication infrastructure. It aims at facilitating and enabling transparent and location–independent invocation of services distributed across heterogeneous computing platforms, regardless of operating system and implementation language. This interoperability is gained through the use of the Interface Definition Language (IDL). All components of the architecture specify their functionality by defining an IDL interface. IDL is a non–algorithmic, descriptive specification language with object-oriented features and support for distributed computing. The IDL interface of a component is mapped to the client's implementation language which provides local proxies for handling argument and return value conversion and ORB–access. These proxies

[3]**R**equests for **P**roposals stage issued by the OMG to its members and interested parties.

Static invocation interface (SII)	For each server interface, one client side stub is generated. The SII referrs to these stubs.
Dynamic invocation interface (DII)	IDL interface to generate request dynamically. (Client side)
Dynamic skeleton interface (DSI)	IDL interface to generate upcalls into server code dynamically. (Server side)
ORB interface	IDL interface for handling initialization, simple naming service, and ORB administration.
Object Adaptor (OA)	Generates object references, handles upcalls into server.
Implementation Repository	Stores information about server implementation.
Interface Repository (IR)	Stores interface definitions of all CORBA objects and provides functions to populate and query the IR.

Table 1: CORBA features and their use.

are generated by a stub–compiler, according to a standardized language mapping, distinct for each programming language. Standardized language mappings for IDL to C, C++, SmallTalk and Ada are available [Obj95b]. Mappings for Java, Cobold and other languages are in preparation.

From a very abstract point of view, can the entire OMA–framework be seen as a distributed system consisting solely of interacting and collaborating *CORBA objects.*[4] Each of these objects is identified by a unique 'id', more formally referred to as object reference. A

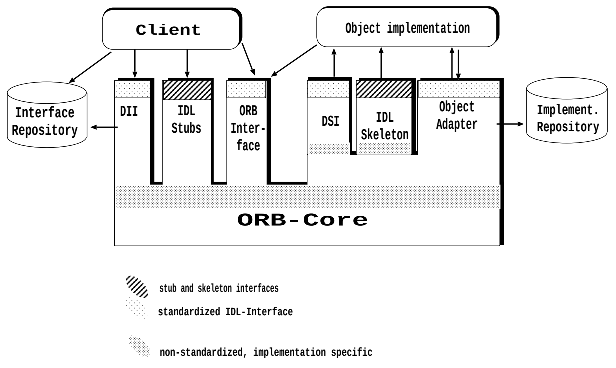


Figure 3: The Common Object Request Broker Architecture (adapted from [OMG 95]).

detailed architecture of the ORB is presented in Figure 3 and briefly described in Table 1. The following sections will go into more detail and show how the individual features of the architecture are used in the open GIS design.

A more detailed description of the CORBA architecture can be found in the OMG specification [Obj95b] or in a surveying article [Vin97].

[4]All components, interacting via the ORB, i.e. CORBA services, CORBA facilities, and domain applications are denoted as 'CORBA objects' in the OMG terminology. All CORBA objects derives from a common supertype, the `CORBA::Object` interface which provides a common set of operations.

GIS requirement	**CORBA**
Component integration	IDL, Object Model
Dynamic extensibility	DII/DSI, IR, Trader
Self describing	IR, Object Model
Software engineering	IDL, Object Model
Scalability,	Encapsulation
fault–tolerance, performance	not directly addressed

Geo–data imposed	**CORBA**
data–volume	weak in CORBA 2.0
proprietary standards	CORBA–GIS–facility
complex data model	not directly addressed
data reuse	not directly addressed
extensive user interaction	supported partly by DII/DSI

Table 2: Mapping of CORBA features and Services to open GIS requirements.

4.2 Synergy of open geographic information systems and CORBA

In this section we demonstrate how to obtain the above identified requirements of open GIS by realizing them with CORBA features and Services. Table 2 presents a summary of the requirements we address and shows how they are implemented with CORBA. We emphasize the first three requirements since we feel, that they are not only essential for open GIS design, but also for distributed system design in general.

Component integration

As outlined above, open GIS are composed of different specialized components, which are seldomly designed with cooperative and integrative goals in mind. These components might be highly specialized libraries or entire sub–systems which might have been previously developed and used in an altogether different context. CORBA IDL provides flexible and extensible support for defining interfaces for such components. IDL is tailored towards object-oriented languages but does not exclude other paradigms which is demonstrated by the fact that mappings for language like Cobold, Lisp, and TCL have been independently proposed. To integrate legacy code into the CORBA framework the application designer must define an IDL interface for the part of the code she wants to expose to the user. A stub–compiler maps the interface definition into stubs and skeletons which are linked together with the application on client and server side, respectively. Depending on the nature of the legacy code/system a thin layer of wrappers must additionally be provided, by the applications designer, to appropriately interface the stubs to the legacy code. This is demonstrated by an example in Figure 4.

Dynamic extensibility

The ability of a system to discover and use new components at run–time, as they become available, is referred to as *dynamic system extensibility.* This is an important feature for distributed systems, since it introduces a very high degree of flexibility, basically permitting every component to call on every other component without prior to system–compile–time–knowledge of each other. This is especially useful in domains, as dynamic as the Internet, where services become and cease to be available, for a variety of reasons, on a very rapid and nearly uncontrolled basis. Moreover, it is undesirable to recompile an entire distribute system each time a new component is added, since recompilation implies system down times. Furthermore, recompilation might not even be possible due to the distributed nature of the application.

```
module GIS2 {
    interface
Spatial_Allocator{
      exception
nServerDown{string msg};
      ...
      Map
allocate_location(Map m1,        □
Map m2)
        raises(ServerDown);
      ...
    ...};
};
```

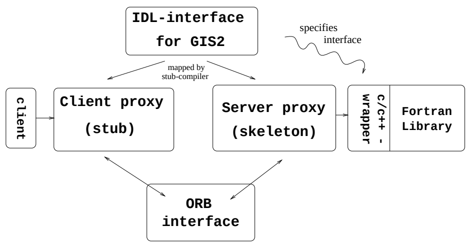


Figure 4: IDL interface for component integration and proxy pattern.

This kind of flexibility in a distributed system demands for a highly sophisticated infrastructure which supports the automated discovery of components, the run–time accessibility of their interfaces, and means to dynamically invoke operations defined in the interface. CORBA supports all of these features.

CORBA Services, like the *Naming*, the *Event*, and the *Trader* service, more fully described below, offer functionality for components to discover other components by name and by property attributes. The CORBA *Interface Repository* (IR) stores the interfaces of all CORBA objects on the object bus. The *dynamic invocation interface* (DII) on the client side and the *dynamic skeleton interface* (DSI) on the server side furnish the necessary functionality to generate request dynamically.

Dynamic extensibility is very useful for drag–and–drop and visual programming environments, or network browsers where at compile time interface information of certain components may not be available, since these components are either non–existing or not yet discovered or even needed.

For the sake of the example, assume that in the geomarketing application from above, different market analysis services may be used by our application over the network. The functionality these services provide all derive from a standardized interface defined by a CORBA facility (see section 4.4 for details on facilities). Below we will address how to discover components, for now we assume that we already have the object reference. Given the object reference a client uses the DII interface to generate a request. First it retrieves the component's interface from the IR by invoking the '`get_interface`' operation on the reference. This will return an object representing the component's interface. The IR–IDL–interface provides operations to extract the signature of the operation to be invoked from this object[5]. A request is simply generated by populating a '`request object`' with the argument values and the operation name.

[5]The client needs to either know the operation it wants to invoke or it must be provided to him through user input (drag–drop user interaction). We assume the former in this example. The market research institute's interface must conform to a CORBA facility of this domain which our client knows.

Self-describing components

A component is self-describing if it is capable of exposing its behavior, i.e. interface, at run time to other participants. As we have seen above, this is important for generating requests dynamically. It is also useful for describing interfaces to interacting users, upon demand, as we will see below. CORBA supports these '*introspective capabilities*' by keeping metadata, i.e. the IDL defined interfaces of all CORBA objects in its IR, accessible for all CORBA objects. Furthermore, all CORBA object subtype implicitly from a common supertype, defined by the `CORBA::Object` [6] interface, which provides several operations callable on all object references in the system. One of these operations, '`get_interface`', returns an object reference to an object in the IR describing the current object's interface.

In our example this is used to expose service attributes and operations to interacting users. E.g. a new service icon appears on our display which indicates the availability of a prior unknow market research service. Note, that our application is linked to an event channel which forwards automatically events to us which we are subscribed to (see Event Service below, for further details on how this works). The user may select the icon which then displays its interface, showing attributes and operations. The user may chose to use this service and pass it the map, under consideration, for further processing. Clearly, map types, as indicated by the operation's '`in`' argument modes must conform.

Software engineering requirements

The software engineering and operational requirements are rather general in nature and apply to all kinds of software systems. Below we address the support CORBA offers to obtain them.

Source code portability of an applications designed against CORBA APIs, is to a large extend possible given the CORBA 2.0 specification and its various implementations. Figure 4.1 shows which parts of the CORBA specification are not fully complete and therefore left for the vendor to design in a proprietary manner. This of course introduces problems when switching to a different vendor's ORB. Much of these portability issues are leveraged through the use of IDL and the product specific IDL–stub–compilers, shipped with every product. On the server side, however, several decisions are left open for the vendor to complete. The upcoming CORBA 3.0 recognizes this problem and promises a refined standard.

The OMG CORBA specification does not dissallow vendor specific additions, as long as the full functionality of the specification is implemented, an ORB is considered to be fully CORBA–compliant. Using such 'value–adds' in an application, clearly renders application source code portability impossible.

Code reusability and extensibility, partly addressed in the above sections are clearly consequences of the object-oriented character of the underlying object model.

Fault–tolerance and performance are not addressed by the current CORBA 2.0 standard and are highly product specific. An OMG working group on real–time CORBA has been assembled which will take up these issues.

[6]'Implicitly', means that the inheritance is not explicitly indicated in the interface of a CORBA objects, but automatically performed by the system.

4.3 Using CORBA Services in the open GIS design

The *CORBA Services* aim at supporting the application developer with common tasks such as finding object references by name, or by properties, offer persistent storage for objects, as well as transactional and messaging support. Altogether 15 services have, so far, been standardized by the OMG [Obj95c]. We are not aware of a single CORBA implementation which supports all of them, but we expect to see a complete set of services emerge in the future. Below we discuss two services which support the discovery of components in the distributed system.

CORBA Event Service

The CORBA Event service specifies interfaces for asynchronous communication between objects. Abstractly speaking, an *event* is a change in state in one object which could be of interest to other objects in the distributed system. This could, for example, be used to notify a client, indicating availability of a server, or simply signaling an erroneous condition in one part of the system.

The Event service supplies an *event channel* which decouples the event supplier from the event consumer[7]. Any number of suppliers and consumers can register with one or more channels. In general, the 'parties', involved, will not know of each other[8]. The events exchanged and queued in the channel may either be untyped, in which case a common event semantic must be agreed upon, or typed. The latter case lends itself well to a selective event notification and dissemination scheme where a consumer is only interested in events of a certain type, as it is the case in the geomarketing example where the system is only interested in the appearance of new market studies.

As indicated in the previous section this mechanism may be used to automatically display a message on the user–interface, signaling the availability of a new market analysis service. We assume that the geomarketing application has registered with an event channel announcing its interest in marked–study–services. Upon receiving such an event, the event channel pushes it further to all the interested parties, among them our application.

CORBA Naming and Trader service

In order for clients to access services in the CORBA framework, object references to these services are required. Both the CORBA Naming and Trader Service provide this functionality. The CORBA Naming Service administers a mapping of names to object references. It permits objects to query for references given names. Servers may register their object references and names with it. The Trader service, on the other hand, allows objects to be discovered based on a list of properties published by servers and requested by clients. A property is a named–value pair, like `(cost--per--transaction, cost)`. A server registers itself with the Trader service by passing it a list of properties, a service type and its object reference. A service type is the server's interface to its operations. A client queries

[7]Event supplier and consumer are terms used by the OMG to refer to objects which generate and consume events. respectively.

[8]Special functionality is supported by the service which allows for a supplier and consumer to directly address each other via the channel.

the Trader by passing it a list of properties. The Trader returns an object reference of a matching server, if one exists. Using dynamic invocation the client may invoke operations on the server.

4.4 Towards a CORBA–GIS–facility

In the above discussion we have focused on the use of CORBA in the design of an open distributed system in general. Naturally, several of the requirements imposed by geo–data on the design of an open GIS are not addressed by CORBA, since its objectives are more general. Domain specific needs are incorporated in the OMA architecture through IDL defined frameworks instantiated as domain specific CORBA facilities.

It comes to our surprise that the OMG, has so far not addressed the IT needs of the geographic domain by issuing a working group on the subject. CORBA facilities for the medical, financial/accounting, telecommunications and other domains have been discussed and requests for information to interested parties have been issued.

In the sequel, we take therefore a closer look at the benefits and functionality such a *CORBA–GIS–facility* should offer. We base our discussion on the requirements of open GIS imposed by geo–data peculiarities, as described earlier.

The benefits of such a facility are obvious. A wide audience would profit from a closer integration of GIS needs into the OMA architecture. For instance, standardization bodies, like OGIS would have a well defined setting within which to operate. Currently, OGIS invests lots of effort in defining a generic GIS model, interoperable with all kinds of distributed computing platforms (DCP). As we will see below in Section 4.5, the large difference in features supported by the individual DCPs imposes severe restrictions on the design of a generic geo–processing–model. GIS application developers and vendors would have standardized interfaces, not only encapsulating their special needs, but also allowing them to easily use and reuse other components of the infrastructure. Users would benefit from the uniform interface provided by the facility.

We see a CORBA–GIS–facility providing the following functionality:

- Measurable geo–data representation formats.
- Geo–data visualization and user interaction support.
- Map representation and query support.
- Complex data modeling functions.

All GIS applications process geo–data in one form or another. A standardized representation format, for the data in general, gives rise to flexible application interoperability. For 'measurable data', like temperature, density or weight, this format needs to account for canonical representations of measurement values, conversions between different units, and extensible and typed unit computations. For maps, another very common type of geo–data, standardized formats specifying representation, spatial query operations, and visualization means need to be present. This could be achieved using a concept such as a the map–windowing technique for editing and spatially querying maps proposed in [Voi95]. In addition, map overlaying, incorporation of location–dependent data and map transformations should be supported. Our geomarketing example underlines these needs particularly

Issue	**CORBA**	**OLE/COM**	**ILU**
Organization	OMG [Obj95b]	Microsoft [Bro96]	XeroxParc [JS97]
Standard	open	proprietary	not a standard
Languages supported	C/C++, ADA, SmallTalk,Cobold	C/C++	C/C++, LISP, Modula–3, Python
Operating System	all common ones	Microsoft's	all common ones
Paradigm	objec oriented	object–based, i.e. no inheritance	object oriented
Open GIS Design Requirement support:			
component integration	supported (via IDL, IR)	possible for C/C++ components only	supported (via ISL/IDL)
dynamic extensibility	supported	supported	not supported
self–discribing	supported	supported	supported
scalability	distribution, federation, possible	across adr. spaces only; expected in DCOM	distribution possible
fault–tolerance	not addressed	not addressed	not addressed

Table 3: Comparison of distributed computing platforms w.r.t. open GIS design requirements.

well. Complex data modeling functions could be defined in a module based on the OGIS specifications.

4.5 Alternate infrastructures

In this section we briefly look at alternate distributed computing environments and compare them with the CORBA infrastructure. Our comparison focuses on the support the individual systems offer for the open GIS design requirements identified. Table 3 depicts the comparison. Due to limited space can we not go into detail about the individual architectures. We therefore try to give a brief overview of the different systems in the first part of the table.

From this comparison we see that CORBA offers important benefits over alternate approaches. CORBA is available on a large range of platforms, it supports most of the fundamental design requirements of open GIS, and, as pointed out in the introduction, offers bridging support to interoperate with other architectures.

5 Conclusion

An open GIS is a generic framework for applications processing geographic data. The idea is to transparently integrate several highly specialized sub–systems into one coherent overall system. In particular, due to the nature of geographic data–processing (e.g. distribution of data sources) and the increasing trend towards software component reuse will these systems be mostly distributed.

In this paper we analyzed the requirements of open distributed systems in general and open geographic information systems in particular. We proposed a generic architecture for open distributed geographic computing. We instantiated this architecture with a particular example application borrowed from the geomarketing domain. We studied the potential of CORBA as the underlying communication infrastructure in this design.

We gave a detailed description of how CORBA intervenes in the overall system design and pointed out its strength and weaknesses. From this study we concluded that CORBA proves excellent as underlying communication infrastructure. Its limits lie in constraints imposed on the open GIS design by the peculiar nature of the geographic domain. We tried to leverage these limits by proposing a general outline of a CORBA–GIS–facility which adds domain specific services to the OMA/CORBA framework. We believe that a firm understanding and specification of this facility will be of great importance in the near future.

We are currently working on a prototype open GIS implementation based on the architectural design principles outlined throughout the paper. The components we are integrating are the GIS GRASS [GRA], the DBMS O_2 [BDK92], and the geometric algorithm library LEDA [MN90]. The user interface is based on the graphical library IlogViews [Ilo93]. We are using IONA's implementation of CORBA as underlying communication infrastructure. We are also working on a formal specification of the CORBA–GIS-facility which will serve as a proposal for the OMG to also address the geographic information processing domain with its standardization efforts.

Acknowledgments:
We wish to thank Andreas Meissner who provided us with the user interface screenshot of the prototype.

References

[AAD+93] R. Ahmed, J. Albert, W. Du, W. Kent, W. Litwin, M, and C. Shan. An Overview of Pegasus. In *Research Issues in Data Engineering*, pages 273–277. IEEE Computer Society Press, 1993.

[AKC94] D. Abel, P. Kilby, and M Cameron. A Federated Systems Approach to the Design of Spatial Decision Support System. In *Spatial Data Handling*, pages 46–59, 1994.

[Ama97] B. Amann. Integrating GIS Components with Mediators and CORBA. Technical Report 97-09, Cedric-CNAM, CNAM, Paris, 1997.

[BDK92] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object-Oriented Database Systems: The Story of O_2*. Morgan Kaufmann, San Mateo (CA), 1992.

[BR94] B. Bruegge and E. Riedel. A Geographic Environmental Modeling System: Towards an Object-Oriented Framework. In R. Pareschi M. Tokoro, editor, *Proc. 8th European Conf. on Object-Oriented Programming*. LNCS No. 821, Springer-Verlag, 1994.

[Bro96] K. Brockschmidt. *Inside OLE*. Microsoft Press, 1996.

[CI-96] CI–Labs. OpenDoc. available at `http:www.cilabs.org`, 1996.

[Dan94] J. Dangermond. ArcView Status and Direction. *ESRI Arc News*, 16(2):1–6, 1994.

[GMS+97] O. Günther, R. Müller, P. Schmidt, H. Bhargava, and R. Krishnan. MMM: A Web-Based System for Sharing Statistical Computing Modules. *IEEE Internet Computing*, 1(3), 1997.

[GRA] Geographic Resources Analysis Support System. Information available at: http://softail.cecer.army.mil.

[Güt94] R. H. Güting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4), 1994.

[Ilo93] Ilog. Ilog Views Version 1.1, Reference Manual, 1993.

[JS97] B. Janssen and M. Spreitzer. ILU Reference Manual. Technical report, Xerox Parc, 1997.

[KKN+96] A. Koschel, R. Kramer, R. Nikolai, W. Hagg, and J. Wie sel. A Federation Architecture for an Environmental Information System Incorporating GIS, the World Wide Web, and CORBA. In *Proc. Third International Conf. on Integrating GIS and Environmental Modeling*, 1996.

[LT92] R. Laurini and T. Thompson. *Fundamentals of Spatial Information Systems.* The A.P.I.C. Series, Number 37. Academic Press, 1992.

[Mei97] A. Meissner. Data Visualization in Open GIS. Master Thesis, Freie Universität Berlin, November 1997), 1997.

[MGR91] D. J. Maguire, M. F. Goodchild, and D. W. Rhind, editors. *Geographical Information Systems: Principles and Applications.* Longman Scientific and Technical, London, 1991.

[MHG+92] F. Manola, S. Heiler, D. Georgakopoulos, M. Hornick, and M. Brodie. Distributed Object Management. In *Intl. J. of Intelligent and Cooperative Information Systenms 1, 1*, 1992.

[MMS+96] E. Mesrobian, R. Muntz, E. Shek, S. Nittel, M. LaRouche, and M. Kriguer. OASIS: An Open Architecture Scientific Information System. In *Sixth International Workshop on Research Issues in Data Engineering - Interoperability of Nontraditional Database Systems*, pages 107–117. IEEE Computer Society, 1996.

[MN90] K. Mehlhorn and S. Näher. LEDA: A Library of Efficient Data Types and Algorithms. In *ICALP 90 Automata, Languages and Programming*, pages 1–5. Springer, 1990.

[Obj90] Object Management Group. The Common Object Request Broker Architecture and Specification. Revision 2.0. Technical report, OMG, 1990.

[Obj95a] Object Management Group. *CORBA Facilities.* OMG, 1995.

[Obj95b] Object Management Group. The Common Object Request Broker Architecture and Specification. Revision 2.0. Technical report, OMG, 1995.

[Obj95c] Object Management Group. The CORBA Common Object Service Specification. Technical report, OMG, 1995.

[Obj96] Object Management Group. CORBA–OLE Bridging. Technical report, OMG, 1996.

[OGI96a] OGIS Technical Committee and the Open GIS Consortium, Inc. The Open GIS Abstract Specification, 1996. Available at `http://www.ogis.org/public/abstract.html`.

[OGI96b] OGIS Technical Committee and the Open GIS Consortium, Inc. The Open GIS Guide (Part I): Introduction to Interoperable Geoprocessing, 1996. Available at `http://www.ogis.org/guide/guide1.htm`.

[OSF96] OSF. Distributed Computing Environment. Technical report, Open Software Foundation, 1996.

[PAGM96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object Fusion in Mediator Systems. In *Intl. Conf. on Very Large Databases (VLDB)*, 1996.

[RAH$^+$96] M. Tork Roth, M. Arya, L. M. Haas, M. J. Carey, W. F. Cody, R. Fagin, P. M. Schwarz, J. Thomas, and E. L. Wimmers. The Garlic project. In *Proceedings of the 1996 ACM Intl. Conf. on Management of Data (SIGMOD)*, page 557, 1996.

[TIN93] TINA–C. DPE Phase 0.1 Specification. Technical report, Telecommunication Information Networking Architecture Consortium, 1993.

[Vin97] S. Vinoski. CORBA Integrating Diverse Applications Within Distributed heterogeneous Environments. *IEEE Communications Magazine*, 14(2), 1997.

[Voi95] A. Voisard. Mapgets: A Tool for Visualizing and Querying Geographic Information. *Journal of Visual Languages and Computing, Academic Press*, 6, 1995.

[VS98] A. Voisard and H. Schweppe. Abstraction and Decomposition in Open GIS. *To appear in Intl. Journal on Geographical Information Systems (IJGIS)*, 1998.

[WWC92] G. Wiederhold, P. Wegner, and S. Ceri. Toward Megaprogramming. *Communications of the ACM*, 35(11):89–99, 1992.