CORBA–BASED INTEROPERABLE GEOGRAPHIC INFORMATION SYSTEMS

H. – ARNO JACOBSEN
Humboldt–Universität zu Berlin
Institut für Wirtschaftsinformatik
D–10178 Berlin, Germany
jacobsen@wiwi.hu-berlin.de

AGNÈS VOISARD
Freie Universität Berlin
Institut für Informatik
D–14195 Berlin, Germany
voisard@inf.fu-berlin.de

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.

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.

Keywords— **Corba, open GIS, distributed geographic application, distributed information system.**

I. INTRODUCTION

A GIS enables to input, store, query, analyze, display, and select spatial (geometric, topological) data, as well as non-spatial data [1], [2]. 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. First of all, 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. Second, 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 for a routing application.

It is clearly more efficient to utilize an expert system or a powerful geometric algorithm library than to re–develop it within the system. 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.). 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 and their steadily growing performance. 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.

This research was supported by the German Research Society (DFG), SFB 373/A3.

As a consequence, open GIS allow to handle applications that could not be handled by traditional GIS, or were barely handled by them. 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 reliance on 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; (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 distributed computing infrastructure, in this case CORBA. This involves considering the following aspects: (1) cooperation of heterogeneous components, (2) component interoperability, (3) component integration, (4) dynamic extensibility.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents a requirement analysis of open GIS. Section IV outlines the benefits CORBA offers for open GIS design. We conclude by arguing for the closer integration of open GIS into the Object Management Architecture, as an additional domain framework.

II. RELATED WORK

One task in the realization of open GIS is the seamless integration of data. In the database area much work has gone into developing methods for integrating data from heterogeneous sources. The architectures of such systems is usually centralized around a mediator which communicates with wrappers. Open GIS design goes beyond data source integration, but benefits greatly 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), and shortest path algorithms. Recent work on the definition of a framework to call software packages remotely (e.g., [3]), 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 control on the overall system.

Regarding geodata exchange and standardization, the work performed, over the past few years, by standardization bodies, such as NIMA and ISO/TC287 in the US, CEN/TC211 in Europe, SAIF in Canada, and Interlis in Switzerland must be noted. Significant work has been carried out by the Open GIS Consortium in form of specifications. The Open Geodata Interoperability Specification (OGIS) [4], for example, 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 [5] and the OGIS Services Model. However, to the best of our knowledge, the mapping onto an abstract DCP, which is one of our primary concerns, has not been studied in depth in this context.

One of the goals of our work is to demonstrate how CORBA [6] 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 [7] or Microsoft's OLE/COM [8]. OLE/COM [8], which is merely 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 [9].

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 [10], [11]). Our approach is different as we focus on services rather than on data integration. [12] 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 [13], 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.

III. OPEN GEOGRAPHIC INFORMATION SYSTEMS

This section identifies the requirement of open GIS. We describe the architecture of a generic open GIS and discuss its main components. We refer to such a system as an "overall system" composed of participating (sub–)systems, or components.

A. Requirement analysis

An open GIS aims at providing a single coherent framework that appears to its user as one overall system. Each participating system specifies its interface in a common interface definition language, enabling mutual use. The overall system provides one consistent interface shielding the user from the various interacting and cooperating components.

This goal in mind and given the observations listed in the introduction, we can characterize the requirements of open GIS as follows. It is important to note that most of these requirements will also apply to distributed system in general. GIS specifics are summarized thereafter.

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 in–house. 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.

Dynamic extensibility: The system must allow the discovery of new components at run–time. It must 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 without halting the entire processing.

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.

Software engineering requirements: From a software engineering point of view portability is a key requirement, since the framework must be easily adaptable to different application domains.

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, become more important.

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

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:

Data volume: The need to handle huge volumes of data (MBytes for map, GBytes for atmospheric data). This prohibits the use of classical techniques (e.g., for data exchange and storage).

Data representation: 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.

Complex data models: Support for *complex* data models due to highly structured data and to specific data–bound operations. These include the afore–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.

Reuse of data: The growing interest in, and need to, reuse data around the world, which puts constraints on data

[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.

collection, representation and storage techniques.

GUI considerations: Extensive user interaction in handling the different data sources, due to the visual and hyper aspects of geospatial data.

B. Architecture of a generic open GIS

Prior to presenting the architecture itself, let us describe an illustrative example. A whole family of applications which are GIS related, as for instance, collaborative spatial decision making and geomarketing applications, will greatly benefit from the integrated approach of the new GIS generation. The typical pattern in this type of applications is: (i) visualize a geodata set (usually regional maps[2] together with addresses), then (ii) overlay sectorial information, and then (iii) answer a precise question or check hypothesis in order to draw conclusions. A detailed discussion relating this example to the issues raised here can be found in [14].

Figure 1 shows the architecture of the generic open GIS. For the sake of simplicity, only the main modules are depicted. The system is centralized around the *integrator*.

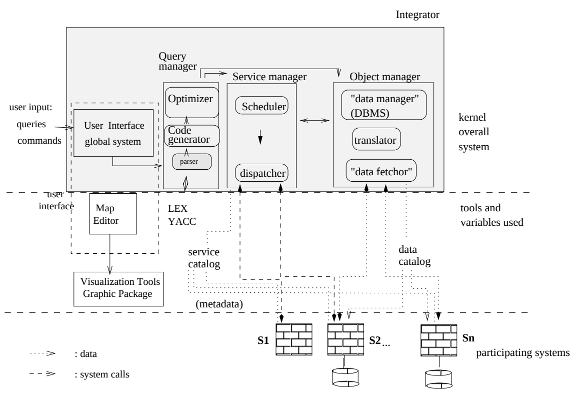

Fig. 1. The generic open GIS architecture

The user interacts with the system via the GUI which communicates directly with the *query manager*. Maps can be displayed by the user; queries are entered in the *query editor* using a spatial query language. All available services and data, together with basic operations on them, can be visualized via popup menus.

The *integrator* has a role similar to a mediator. It dispatches the user instructions to the subsystems and integrates their results. It is composed of a *query manager* (QM), an *object manager* (OM), and a *service manager* (SM). The QM is composed of a code generator and an optimizer. It dispatches the appropriate queries to the object manager and the service manager. The OM is in charge of fetching relevant objects from repositories, as well as defining and storing them locally. The SM is in charge of communicating with all services provided by participating systems. It is mainly composed of a *scheduler* and a *dispatcher*. It draws on the *service catalog* to further dispatch operations.

Data and metadata: In the overall system, both alphanumeric data and geospatial data coexist. Metadata on both kinds of data plays a crucial role, as it facilitate search, browsing, and data integration. It describes the contents of geospatial and alphanumeric data collections and is stored in the *catalog*. This includes type information, location, access rights, owner, and creation date. The metadata is applied to find out which resources are present in the underlying data–sets and in 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 individual operational components are concerned, metadata related to their services, how to start and use them, as well as location information have to be stored. This information is exported by the participating systems during a check–in procedure.

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.

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. The next section concentrates on CORBA and on how it can be used to design such open geographic information systems.

IV. 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 [6], OLE/COM [8], Tina–C [15] or DCE [7]. 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. 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.

A. The Common Object Request Broker Architecture (CORBA)

CORBA is an interoperability platform for building distributed applications, specified by the Object Management Group [16], a consortium of more then 600 member institutions (industrial, as well as academic). It aims at leveraging interoperability problems that arise from the integration of systems in heterogeneous distributed environments (operating systems, network protocols, programming languages, etc.).

B. Synergy of open GIS 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 I and Table II present a summary of the requirements we address and show how they

[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.

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

TABLE I

Mapping of CORBA features and Services to open GIS requirements.

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 II

Support for GIS data imposed requirements.

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 Cobol, 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 s/he 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 in Figure 2 and Figure 3, respectively.

```
module GIS2 {
   interface Spatial_Allocator{
      exception ServerDown{string msg}

      Map allocate_location(Map m1, Map m2)
      raises(ServerDown);
      ...
   } // interface
} // module
```

Fig. 2. IDL interface exemplifying component integration.

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

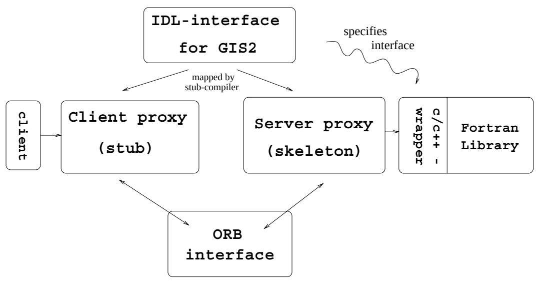

Fig. 3. Proxy pattern.

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.

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. Below we 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 returns 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[3]. A request is simply generated by populating a '`request object`' with the argument values and the operation name.

Self-describing components: A component is self-describing if it is capable of exposing its behavior, i.e., in-

[3]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.

terface, 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` [4] 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. For example, 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 2.2 specification and its various implementations. The specification is unfortunately not fully complete and leaves minor design decisions open. This introduces portability problems when switching to a different ORB. Much of these portability issues are leveraged through the use of IDL and the ORB specific IDL–compilers, shipped with the ORB. On the server side, however, several decisions are left open for the vendor to complete. The upcoming POA (Portable Object Adapter) recognizes these problems and presents a refined solution for object implementation integration.

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 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.

C. 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 [17]. 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 especially useful for our application.

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 signal 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[5]. Any number of suppliers and consumers can register with one or more channels. In general, the 'parties', involved, will not know of each other[6]. 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 user 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 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.

D. 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

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

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

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

through IDL defined frameworks instantiated as domain specific CORBA facilities.

The OMG, has just recently started to become interested in the geographic domain by starting a SIG (special interest group) on the topic [7].

In the sequel, we take therefore a closer look at the benefits and functionality such a *CORBA–GIS–facility* could 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. Standardization bodies, for instance, 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). The large discrepancy in features inherent to the individual DCPs imposes severe restrictions on the design of such a generic geo–processing–model. Such a model, if not specifically tailored to each individual DCP, would be too generic to be practically useful. On the other hand, a specific mapping of the model onto each DCP would certainly not enhance application portability.

A CORBA–GIS–Facility would give GIS application developers and vendors a set of 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 simple CORBA–GIS–facility providing at least the following functionality: (1) Measurable geo–data representation formats. (2) Geo–data visualization and user interaction support. (3) Map representation and query support. (4) 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 (raster) maps, the most 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 [18]. In addition, map overlaying, incorporation of location–dependent data and map transformations should be supported. Our geomarketing example underlines these needs particularly well. Complex data modeling functions could be defined in a module based on the OGIS specifications.

V. 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.

[7]When we started to become interested in the combination of GIS with CORBA technology, in late 1996, no such efforts where underway.

References

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

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

[3] 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.

[4] 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`.

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

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

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

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

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

[10] 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.

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

[12] 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.

[13] 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.

[14] H.-A. Jacobsen and A. Voisard. CORBA-Based Interoperable Geographic Information Systems, 1998. ICSI Berkeley Technical Report No. 98-011, April 1998.

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

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

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

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