View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by KITopen

A CORBA-based Approach to Data and Systems Integration for 3D Geoscientific Applications[1]

Oleg T. Balovnev, Andreas Bergmann, Martin Breunig, Armin B. Cremers, Serge Shumilov

Institute of Computer Science III

University of Bonn

Tel: +49 228 73-4535

Fax:+49 228 73-4382

e-mail: martin@cs.uni-bonn.de

Abstract

Today's geo-information systems are not open enough to provide a required level of data and systems integration. In this paper we introduce an approach to data- and system- integration which is based on the common object request broker architecture (CORBA), which allows for various applications to exchange data as objects. As a result not only data but also data processing methods can be made accessible for the remote applications thus creating a basis for exchange of geo-services. The central role in the proposed approach plays an object-oriented geodata store built on top of common object model: objects which are uniformly stored in the database serve as mediators between extremely heterogeneous representations inherent to different geo-scientific applications. The approach is evaluated by the development of a prototype distributed environment for concrete geological and geophysical 3D modeling systems.

Keywords: open geo-information system, data and systems integration, 3D geo-information system, geo-toolkit, geo-database.

1 Introduction

The data and systems integration issues in geo-sciences have been addressed by [Abe89], [AbW90], [BrP92], [ScW93], [AKD94], [CBR94], [OGI96], [Bre96], [RaL97] and by other authors. However, there are known only a few works investigating object-oriented architectures for the data- and methods exchange between heterogeneous 3D geo-information program components and databases. In this paper we propose a CORBA-based approach [OMG97c] to data- and systems integration. The characteristic feature of this approach is that data can be exchanged between various geo-services as objects. That means that not only data but also data processing methods can be made open for the remote access. The central role in the proposed approach plays a common object model. Objects which are uniformly stored in a shared object-oriented database serve as mediators between extremely heterogeneous representations inherent to different geo-scientific applications.

The approach was applied for data- and system integration in different geo-scientific domains within two projects. The first one deals with uniform access to heterogeneous and distributed sources of paleoecologic data. Multiple German geoscientific groups participate in an effort to investigate the evolution of the biosphere during the last 15.000 years. They collect lots of data describing local characteristics like strata of drillings, samples of pollen, results of chemical and physical analyses, etc. and store them in many different formats and systems according to their special needs. Much additional information can also be found in data collections like the WDC-A in Boulder. The characterization and classification of processes in different ecosystems, that will support detection of local particularities or anomalies accord-

[1] This work is funded by the German Research Foundation (DFG) within the cooperative project „Interoperable geoscientific information systems".

ing to superior characteristics and changes of the atmosphere, can only be achieved, if the whole amount of relevant information is accessible in an uniform way and can be processed with different tools.

The second project deals with 3D geological mapping. Geological maps present the geometry, lithology, age as well as some other characteristics of diverse geological bodies. Today's geological maps are intrinsically two-dimensional. Only a well-trained geologist can restore in mind underlying 3D structures from the analogue maps. The progress in computer sciences makes actual the maintenance of digital 3D geological maps [Sie88]. However, there is no established geo-information system that could be used for the computer-aided design of such maps. The goal of the project is the construction of a consistent geological 3D-model of Southern Lower Saxony from primary geological data and its iterative refinement by using in rotation specialized geological - GOCAD© [Mal92], [GO98] and geophysical - IGMAS [GöL88] tools. Geophysical modeling applies gravimetric and magnetic evaluations of the potential fields to extrapolate the geological information gained at the earth surface into the depth. However, on the initial stages it is not effective enough because of the large variability of parameters under consideration. To reduce the variability the geo-scientific modeling needs a kind of rough cast which can be provided by the interactive geological modeling with GOCAD. The stratigraphic information obtained in the result of geological modeling is further used for the refined computations of densities within IGMAS.

2 Requirements

Though being apparently diverse, these domains appeared to have much in common from the viewpoint of data- and system integration. Due to many heterogeneous data formats and different computer platforms, the data are usually split into isolated partitions and therefore are not available in the whole volume to a particular application. On one hand this scenario supports effectively the specific needs of a concrete geo-discipline, but on the other hand it prevents the utilization of the entire data for the comprehensive analysis. Bridging the gap requires integration both on the logical (data-related) and the technical (system-related) layers. The logical layer precludes the development of interdisciplinary object models, describing the semantics of geo-objects, and mapping techniques that support semantically correct transformations from source formats to the integrated model. This involves mapping of attribute names, unit conversion, coordinate transformation, etc. The technical layer deals with heterogeneous computer architectures, operating systems and programming environments.

Both domains have common requirements to data and systems integration:

- formal definition of geo-objects in 3D space including their meta-information;
- development of interdisciplinary object models;
- database queries on „integrated data";
- data and methods exchange between geoscientific applications;
- formal description of interfaces for the remote access to geo-services.

Geo-scientific data are extremely heterogeneous. They have very complex hierarchical or network structures which are rather optimized for a particular task than standardized. The total integration of such data will need tremendous expenses. However, necessary steps in this direction can be made with less efforts. Therefore several levels of data integration are reasonable.

- *Meta-data level* provides geoscientists with the knowledge what kinds of data are available and where they are located.
- *Source-data level* provides geoscientists with the ability to pose database queries on integrated data sources. Multiple semantic problems beginning from name or scale conflicts to different semantics of diverse data capture methods arise during this integration. This approach precludes a joint processing of data, therefore its prerequisite is a common database schema, where a sub-schema for particular domains can be modelled, for example, as a perspective [CBR94] of the global schema.

In the paper we focus primarily on the source-data level integration in the geological mapping project.

3 Motivation

When providing uniform access to multiple independent sources of paleoecologic data, we have to deal with heterogeneity on the three levels. To connect an arbitrary geo-application to any remote data source, we first have to bridge the gap on the system level between the generally different hardware platforms and operating systems. On the software level we have to deal with the diversity of tools, that are used for data storage and maintenance at the different sites. They range from flat files to sophisticated relational or object oriented database management systems. On the semantic level, we face a great variety of formats, describing the same real world entity in different ways often depending on the primary use of the data.

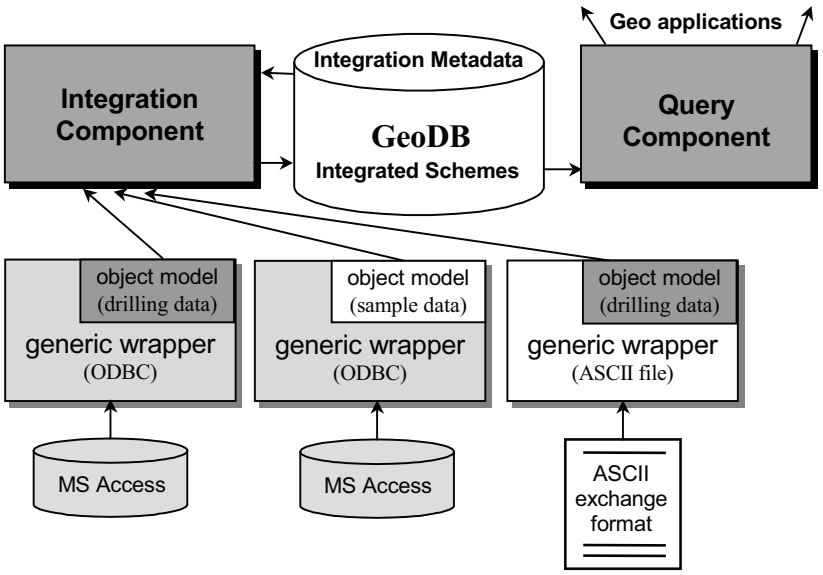

Fig. 2: Integration of heterogeneous geodata sources

Figure 1 shows an example configuration of data sources and the components needed to provide uniform access to integrated geo-data in an open GIS environment. Our goal is to provide integrated access to an arbitrary set of data sources at the same time, that should further on be maintained in a distributed and independent way. We need a set of ”generic wrappers” encapsulating classes of storage tools used at the different sites. These wrappers are coupled with an object model capturing the semantics of the corresponding source data. Note that there is only one wrapper per storage tool class and one object model per data format. At the core of the system we need an integration component, which imports primary data and their semantics from the object models and utilizes certain metadata to guide the integration process. No single database scheme will satisfy the diverse requirements from different disci-

plines. Therefore we will employ techniques like object oriented views in order to manage several integrated schemes in parallel. A query component will provide a uniform interface to access the integrated data from different geo-applications.

As mentioned above a construction of 3D geological maps requires a cooperative application of diverse geological and geophysical modeling tools. In the geological mapping project the geologists located in Bonn use a GOCAD®-based tool while geophysical modeling is carried out in IGMAS® resided in Berlin. Both systems are large historically grown software products which are optimized for their own purposes. Currently the data exchange runs on the ASCII files level with the resulting maintenance problems. In future primary geological data and data elaborated during modeling are expected to be maintained in a database developed on top ODBMS *ObjectStore*. Apart to incompatibilities in data structures and software platform serious problem may occur with the hardware platforms: GOCAD is exploited on SGIs, IGMAS runs on HPs, data are stored on Suns, primary data are partially maintained under Windows NT.

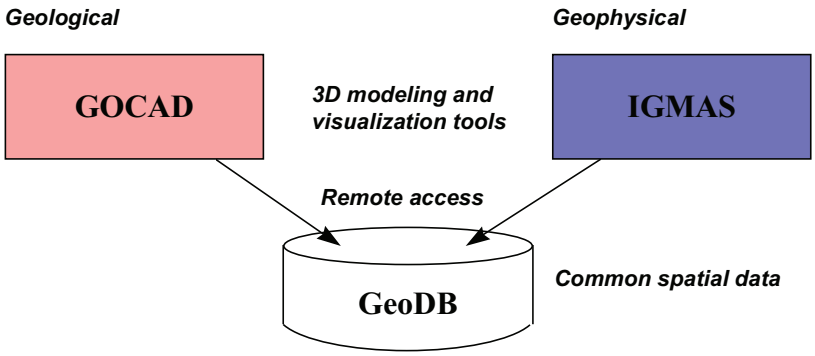

Fig. 1 Remote access to open geodata stores.

Summarizing, within both projects we have the whole spectrum of problems that could occur in data and system integration. The goal is to provide an distributed environment for the free exchange of data between the Geo tools via open object-oriented Geo data stores. In the remainder of the paper we focus primarily on the source-data level integration in the geological mapping project.

4 Data integration

4.1 Multiple spatial data representations

The first step towards interoperation between the geological and the geophysical 3D modeling systems (GOCAD©and IGMAS) was the development of a common object model with the Object Modeling Technique [RB+91]. Geologists operate mainly with geological strata and faults. Geological strata are sheet-like structures in earth with different mineral composition, texture and/or grain size. A fracture in earth materials along which the opposite sides have been displaced is known as a fault. Geometrically a geological stratum is defined as a set of spatial solids. Alternatively a stratum can be modeled as a composition of bounding surfaces: faults and an upper and lower stratigraphic boundaries. Geological and geophysical representations for the same entities may differ both in thematic attributes and representations for geometric and topological data. The diversity of representations arises from the distinct

purposes they serve for. Thus, in IGMAS data structures are optimized for the efficient mathematical computations while in GOCAD classes are oriented toward an efficient visualization. A solid representation is more preferable for geological modeling because it allows an efficient derivation of stratigraphical values for arbitrary locations in space. In GOCAD a solid (referred to as *Tsolid*) can be modeled as a specialized tetrahedra container class which is derived from a general simplicial complex class. The data elements like vertices and constituting tetrahedra are also derived from corresponding atomic geometric classes. IGMAS, on the contrary, exploits the bounding surface representation under assumption that bodies have no holes.

Since both systems were intended to use a *GeoToolKit*-conform database as data store it was reasonable to build the common object model on top of *GeoToolKit*'s class hierarchy. *GeoToolKit* offers classes for the representation of and manipulation with diverse spatial objects within a database. The advantage of *GeoToolKit* for integration purposes is that it supports flexible switches between different spatial representations. The geologist, for example, may choose the tetrahedron network (a simplicial 3-complex) representation for a solid, which can be easily converted (using *GeoToolKit*'s embedded methods) into a bounded surface representation, utilized in IGMAS, and vice versa.

4.2 GeoToolKit

Within the collaborative research center SFB350 at the University of Bonn we have develop a component software called *GeoToolKit* [BBC97], which is intended to facilitate the development of 3D/4D geo-applications. The idea is to provide for the application developer a set of geo-oriented software building blocks involving DBMS-based spatial data maintenance, special support for efficient spatial retrieval, visualization and graphical interfaces, communication, which due to the standardized interfaces an application developer can assemble in a ready-to-use application. A necessary basis for the integration of diverse components is provided by an object-oriented programming environment. Therefore *GeoToolKit* is not a closed GIS-in-a-box package - it is rather a library of C++ classes that allows the incorporation of spatial functionality within an application under development.

GeoToolKit addresses primarily the efficient storage and retrieval of 3D-spatial objects within a database, providing a class hierarchy for the representation of various geometric bodies. Currently *GeoToolKit* includes classes for the representation of simple (point, segment, triangle, tetrahedron) and complex (curve, surface, solid) 3D spatial objects. Complex shapes are approximated and decomposed into a set of adjacent simple objects of the same dimension. Such representation turned out to be especially beneficial for the maintenance of non-regular shapes which are typical for the majority of geo-applications [BBC94b]. Therefore polylines, triangle networks and tetrahedron networks serve as default representation for curves, surfaces and solids, respectively). However, the user can realize his own representations. The *GeoToolKit* class hierarchy is complete: any 3D-spatial object can be modeled either directly by one of the built-in spatial classes or as a composition of these classes within a group. Following the object-oriented modeling technique, real world entities such as drilling wells, geological sections or strata, can be modeled and maintained. Applications developed with *GeoToolKit* simply inherit geometric functionality from *GeoToolKit*, extending it with the application-specific semantics. Spatial objects are maintained in special container classes - *spaces* - which are capable of efficient retrieval of its elements according to their location in space. To provide an efficient retrieval a space utilizes specialized spatial indexes. Geometric operations are closed: operation results can be used in following computations.

4.3 Common Object Model

As a basis representation for geological strata we chose the tetrahedron network representation for a solid (*class gtTetraNet*). A bounded surface representation, utilized in IGMAS, can be easily generated by applying the embedded *getSurfRep* method (designated as derived link). The *gtClosedSurface* class is a specialisation of the standard GeoToolKit *gtSurface* class, extended, in particular, with the *getSolidRep* method used for the tetrahedrization of the hull. Applying this method, a bounding surface representation can be exported in GOCAD.

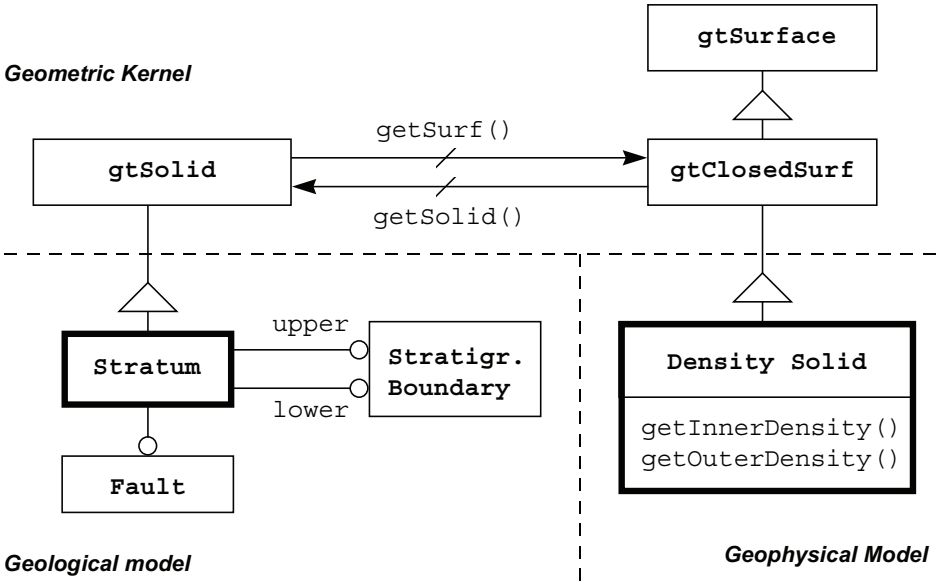

Fig. 2 Common object model for the integration of geophysical and geological sub-models.

The geo-scientific classes *Stratum* and *DensitySolid* are defined as specializations of pure spatial classes *gtSolid* and *gtClosedSurf*, respectively. Naturally they get additional thematic attributes (e.g. lithology or density), relationships (e.g. links to intersected faults), and methods (e.g. functions, providing bi-directional data conversions to/from internal GOCAD or IGMAS representations). However, the geometrical functionality they inherit completely from *GeoToolKit*'s spatial classes.

5 System Integration

5.1 Remote Access to *GeoToolKit*-based Data Stores

Common underlying data structures provide a necessary basis for the successful integration and interoperation of geo-scientific systems built on top of *GeoToolKit*. Shared object-oriented geodata stores serve as mediators for data and services exchange. The most comprehensive and efficient utilization of *GeoToolKit´*s facilities is achieved by a direct database-level client-server communication with these data stores. This approach is preferable but often not realizable, primarily because of the extreme heterogeneity of existing geo-scientific applications and software-/hardware platforms. Many applications are not available for the modifications required for the direct integration with *ObjectStore*. Database-level communication facilities supported within DBMSs generally are not flexible enough for an advanced network navigation. Apart from this, such principal *ObjectStore* architectural feature as entire data processing (including queries) at the client site may lead to an unjustifiable overload of the network making essential a more „intelligent“ access to database objects.

To implement the remote access to a *GeoToolKit*-based data we used initially standard UNIX services: sockets and the remote procedure call (RPC) mechanism. To achieve a higher level of type safety than by straightforward bit-stream data transfer, we implemented data protocols as object classes shared by both client and server. This technique required a large amount of relatively low-level but high-qualified programming efforts. Another drawback is that an introduction of a new data type may require a modification in the control structuresl. In general, these UNIX services are optimized for performance, rather than ease of programming, reliability, portability, flexibility and extensibility. However, the most serious limitation is the lack of generality. A low-level data exchange mechanism turned out to be very efficient for the communication between two particular applications. An involvement of a third application may require a painful re-design of all participants of the communication. What we need is a kind of standard distributed computing platform. Taking into account the object-oriented nature of data in the majority of geo-scientific applications the most suitable solution is *Object Management Architecture* [OMG97] which promises to become a world-wide standard. Basing on *Common Object Request Broker Architecture* (CORBA) we can expect that any other CORBA-compliant application can get an open access to *GeoToolKit*-based data stores.

5.2 CORBA as a Communication Platform

The advantage of CORBA is that it delegates much of the tedious and error-prone complexity associated with the low-level socket-layer programming to its reusable infrastructure. An application only needs to hold a reference to a target object, the *Object Request Broker* (ORB) is responsible for automating other common activities, which usually include locating the target object, activating it if necessary, delivering a request to it, and returning any response back to the caller. A client treats a remote object as an ordinary object utilizing the procedure invocation mechanism conventional for this programming language. Parameters passed as part of the request are automatically and transparently marshaled by the ORB, which ensures correct interoperation between application and objects residing on different platforms.

CORBA object interfaces are described using an Interface Definition Language (IDL). Since the IDL specifications are automatically translated potential inconsistencies between client and server counterparts are significantly reduced, providing a higher degree of type safety than for the bit-stream oriented sockets. For large distributed systems a loss of efficiency inevitable for such universal systems is compensated by the increased extensibility, robustness, maintainability.

```
exception GTK_ObjectExists {};
exception GTK_ObjectNotFound {};
    ...
interface GTK_Space : GTK_Object {
// operations on spatial objects
void insert (in GTK_SpatialObject
    raises (reject);
void remove (in GTK_SpatialObject
    raises (GTK_ObjectNotFound);
GTK_SpatialObject
    retrieve (in GTK_BoundBox bb);
GTK_Space
    intersect (in GTK_SpatialObject
    ...
};
```

```
interface GTK_SpatialObject : GTK_Object
// spatial predicates
boolean intersect (in GTK_SpatialObject
    ...
// spatial operations
GTK_BoundBox getBoundBox ();
GTK_SpatialObject
    intersection (in GTK_SpatialObject
};
interface GTK_Line : GTK_SpatialObject
// functions specific for the lines
    ...
};
```

Fig. 3 Fragments the IDL interface

We re-produced the *GeoToolKit* class hierarchy in CORBA's IDL, trying to keep the interfaces as close as possible to the original C++ ones (Fig. 4). However, the IDL interfaces are significantly shorter, because local efficiency issues essential for the high performance computations in the C++ environment were no longer relevant: the optimization gain is insignificant in comparison with the total expenses for remote procedure calls. Thus, an original *GeoToolKit* spatial object class usually offers a family of functions implementing the same geometric operation which are responsible for the efficient static polymorphic binding. The wrapper does this dynamically without appreciable loss in efficiency.

5.3 Wrapper Architecture

The fact that we have to deal with large data stores which are permanently in use by multiple geo-scientific applications imposes additional requirements on the architecture. First of all, making these data stores CORBA-compliant should not disturb existing applications. Since a database schema evolution may be extremely time-consuming, ideally the CORBA-compliance should be achievable without changing where and how the data are stored. To realize this we followed a database wrapping technique. A wrapper encapsulates the underlying data and mediates between data stores and diverse geo-scientific applications. In our case a wrapper encapsulates access to all *GeoToolKit* classes, providing the full run-time control of their instances. This permits to reuse entire functionality of *GeoToolKit* with minimal expenses.

5.4 Persistency Management

The CORBA standard specifies a flexible distribution model for transient objects. However, we have to deal primarily with persistent objects, maintained within object-oriented DBMSs. These facilities are still not standardized. One of the reasons to choose *Orbix*® as an implementation platform was that it provided a framework for the development of Object Database Adaptors (ODA) involving a special *Orbix/ObjectStore Adaptor* (OOSA). OOSA substitutes the basic object adapter and completely undertakes the management of persistent objects in *ObjectStore* databases. OOSA implements a TIE-approach which is characterized by ... bla-bla-bla.

OOSA helped us to considerably reduce the development costs in the initial stages. However, after several experiments with a rough prototype we implemented on top of OOSA it turned out to have serious limitations and drawbacks. First of all, it was not suitable for the achievement of our main goals - a CORBA compliance without modifications in existing geo-data stores and maximal reuse of *GeoToolKit's* facilities. Therefore we had to re-implement ... Below we present both approaches

Client and server communicate through stab and skeleton counterparts generated from the common IDL specification (Fig. 5). In the approach adopted in OOSA a TIE-object is persistent: it contains a reference to a database object. Because we can not touch a database object interface nor a TIE-object interface (it is automatically generated by IDL-compiler), we have to introduce an additional mediator which task is to provide a correct mapping between CORBA-compliant interface of a TIE-object and a *GeoToolKit*-compliant interface of database objects interface. A mediator converts parameters and forwards the function call to the corresponding database object. When a database operation is completed, the mediator converts the result back in the CORBA-compliant representation and returns the control to the TIE-object. Therefore a request for the creation of a new object results in the creation of the triple of persistent objects: TIE object, mediator object and database object. However, when a

database object is created by a local applications, corresponding TIE and mediator objects do not exist. ... A lazy approach

This approach will work if each spatial object could be referred via an unique name. Geoscientific applications are characterized by a high granularity. They usually consists of en extremely large number of small-scale objects which do not have unique names. However, typically a number of spatial objects is extremely large and they are not maintained as separate database entities with external names. Instead of this they are encapsulated in spatial collections - spaces which typically serve as entry points in spatial databases. A separate spatial object usually is picked indirectly via diverse retrieval functions associated with a space. As mentioned above, our approach benefits from delegating the functionality to *GeoToolKit.* That means that we get selection results in form of database objects. However, for return object to the client we need corresponding TIE objects. The straightforward method was to attach to every database object a reference to its TIE counterpart. However, for the exiting databases this approach implies the modification of the database schema and, consequently, an exhaustive database evolution. To avoid this we maintain a repository of references to TIE. The repository is implemented as a dictionary with a single key equal to the GeoToolKit object identifier. When a TIE object is created it is automatically inserted in the dictionary and vice versa. Having got a GeoToolKit object as a result of spatial query we can always retrieve from the dictionary the corresponding TIE object. Since object identifiers are unique the retrieval from the dictionary is very efficient. For more efficiency the repository is taxonomically partitioned: every particular class of TIE objects maintains its own sub-dictionary.

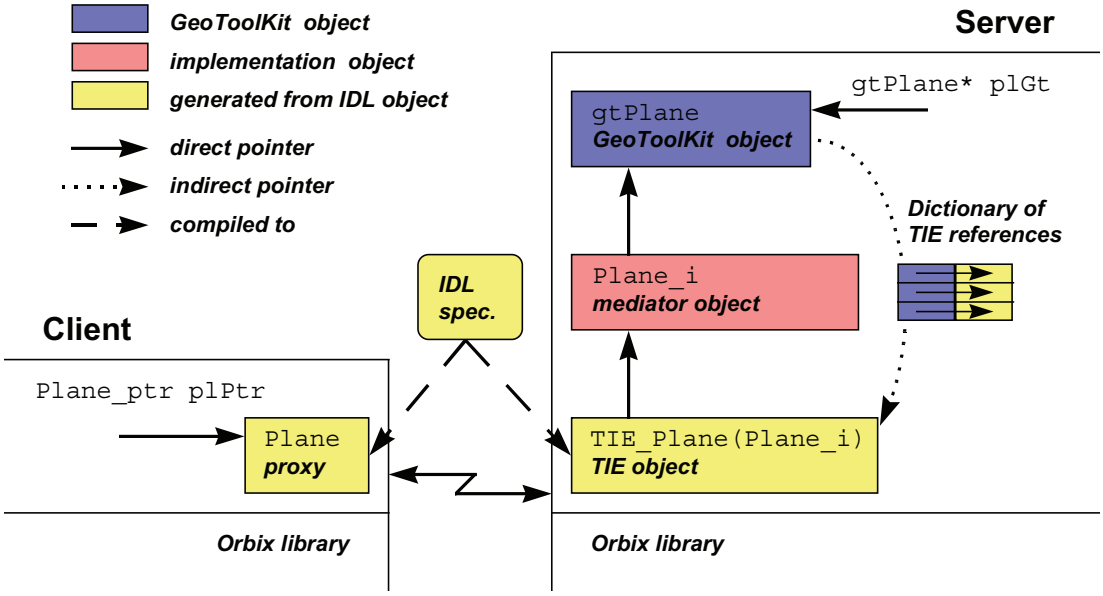

Fig. 4 Wrapper's Architecture.

Thus we achieve a complete separation of pure database and communication components. Therefore when the CORBA-compliance will be no longer needed it can be easily and painlessly abolished. Since the database remains completely unchanged all local applications could continue to work with the data store as before. Moreover, following this method any existing data store at any time can be made CORBA-compliant.

However, there is a potential problem within this method concerning maintaining the referential integrity of the data shared both by local and CORBA-compliant applications. For example, a local application can delete an object, but since it is not aware of the communica-

tion counterparts of the database object, the corresponding TIE and mediator objects will stay in the database. Consequently we get memory leaks which will accumulate with the time. It will not lead to misfunction, however, potentially it will slow down the search for back-references in dictionaries. This problem is characteristic to the object-oriented data model in general since an object can be referred from multiple sources. Since *ObjectStore* has no embedded garbage collector we should implement it explicitly. Usually the corresponding functionality is integrated in the destructor of the removed objects, which sends a corresponding message to the repository manager, which will remove the corresponding entry from the dictionary and delete TIE and mediator counterparts of the database object from the database. This approach requires the total recompilation of all local applications with the additional CORBA-compliant option, though without any change in the source code. periodical total rescanning of TIE-object repositories in order to detect incorrect objects using the metadata support provided within *ObjectStore*. This approach does not require the recompilation of local applications.

We realize this in the lazy mode. The mediator objects are created only when they are requested for the first time from the client.

could hardly provide a mai OOSA forced all mediators to be persistent objects. Taking into account a large granularity of objects characteristic to spatial databases this may result in the significant growth of databases. Dictionaries are completely filled. Therefore a back-reference search becomes much slower. (BBC98). In general, turned out to be not suitable because

CORBA distinguishes between object types that are always passed as reference and non-object types (values) that are copied. However, object types in C++ in general and in GeoToolKit in particular are often used as a convenient aggregation for passing parameters to methods (e.g., in order to reduce the number of parameters). A typical representative of such aggregation in GeoToolKit is a bounding box when it is used as a parameter in various retrieve functions. Obviously, these objects do not need to be represented as independent entities on the server side. Their maintenance on the server site will only lead to the unjustifiable growth of fine-level interactions between server and client. What we need is to convert object's state to a bit-stream on the client side (while passing the parameters) and vice versa - on the server side (while receiving the parameters). However, this conversions should be performed possibly transparent for the user.

Every access to the persistent object must be performed within a transaction. By default OOSA activates a local transaction before each IDL-defined operation on a persistent object begins its execution (if there is no transaction already active). At the end of the operation the OOSA automatically commits the auto-started transaction. Sometimes multiple IDL operations must be performed within a single transaction.

```
#include <GTK_h.hh>
    ...
try {
    // Get the reference to the factory and open database
    GTK_Factory_var gtk = GTK_Factory::_bind (":GTK", host);
    GTK_Database_var db = gtk->openDatabase ("Low-Rhine-Basin");

    GTK_Point_var p1 = GTK_Point::_narrow (gtk->createObject(db, "Point"));
    ... // create three points specifying a vertical plane
    GTK_Plane_var pl = GTK_Plane::_narrow (gtk->createObject (db, "Plane"));
    pl->set_Point (p1, p2, p3);

    GTK_BoundBox_var bb = GTK_BoundBox::_narrow (gtk->createObject (db, "BoundBox"));
    bb->set_Double (250000, 560000, -1000, 254000, 568000, 100);

    GTK_Space_var layers = GTK_Space::_narrow (gtk->findObject (db, "Layers"));
    GTK_Space_var rs = layers->retrieve (bb);
    GTK_SpatialObject_var so = rs->intersection (pl);
    ...
} catch (...) {
    ...
}
```

Fig. 5 Fragments of client-site sources.

An example of the client-site manipulations with spatial objects is presented in the Fig. 5. First of all the client should bind the unique main factory object. This transient object is automatically created by the wrapper both on the server and client sites during initialization. Through the main factory object the client gets the access to a database specified. Usually spaces are named and serve as entry point in the database. Having got a space the client can select a separate spatial object using spatial retrieve methods associated with the space class and specified in the IDL interface. The client can safely convert the retrieved object to the necessary type with the `_narrow()` member function automatically generated for all proxy classes.

6 Conclusions and Outlook

We have presented a new approach to integrate geological and geophysical 3D modelling systems on top of an object-oriented database. Hitherto we have implemented a prototype CORBA-based distributed environment which demonstrates the principles of the remote data and methods exchange between heterogeneous geo-scientific 3D-modelling tools and GeoToolKit-conform data stores. A free data exchange via a common database provided for geo-scientists an opportunity to perform a cooperative adjustment of geophysical density and geological stratigraphic models.

The prototype environment was tested in the local network. In future, however, the performance of data and methods transfer has to be evaluated and optimized in a wide area network.

7 Acknowledgements

The authors acknowledge the excellent cooperation with the groups of Richard Dikau and Agemar Siehl (both Bonn University) and H.-J. Götze (Free University Berlin).

References

[Abe89] D.J. Abel. *SIRO-DBMS. A database tool-kit for geographical information systems.* In: Intern. Journal of Geographical Information Systems, v. 3, pp. 291-301, 1989

[AbW90] D.J. Abel, M.A. Wilson. *A Systems Approach to Integration of Raster and Vector Data and Operations.* In: Proc. 4th Intern. Symposium on Spatial Data Handling, Zürich, V. 2, p. 559-566, 1990

[AKD94] D.J. Abel, P.J. Kilby, J.R. Davis. *The systems integration problem.* In: Intern. Journal of Geographical Information Systems, V. 8, No. 1, pp. 1-12, 1994

[BBC94b] M. Breunig, Th. Bode, A.B. Cremers. *Implementation of Elementary Geometric Database Operations for a 3D-GIS. In Proceedings of the 6th Intern*. Symposium on Spatial Data Handling, Edinburgh, Scotland, pp. 604-617, 1994

[BBC97] O. Balovnev, M. Breunig, A.B. Cremers. *From GeoStore to GeoToolKit: The second step*. M. Scholl, A.Voisard (Eds.): Advances in Spatial Databases, Proceedings of the 5th International Symposium on Spatial Databases (SSD'97), Berlin, Germany, July 1997. Lecture Notes in Computer Science, Vol. 1262, Springer, 1997.

[BBC98] O. Balovnev, M. Breunig, A.B. Cremers. *GeoToolKit: Opening the Access to Object-Oriented Geodata Stores.* M. Goodchild, M. Egenhofer, R. Fegeas, C. Kottman (Eds.): Interoperating Geo-Information Systems, Kluwer, 1998 (in print).

[Bre96] M. Breunig. *Integration of spatial information for geo-information systems,* LNES No. 61, Springer, Heidelberg et al., 171p., 1996

[BrP92] M. Breunig, A. Perkhoff. *Data and System Integration for Geoscientific Data.* In: Proceedings of the 5th Intern. Symposium on Spatial Data Handling, Charleston SC, Vol. 1, pp. 272-281, 1992

[CBR94] A.B. Cremers, O. Balovnev, W. Reddig. *Views in object-oriented databases.* In: Proc. of the Intern. Workshop on Advances in Databases and Information Systems ADBIS'94, Moscow ACM SIGMOD Chapter, May 23-26, Moscow, 1994

[GOC98] GOCAD - http://www.ensg.u-nancy.fr/GOCAD/gocad.html

[GöL88] H.-J. Götze, B. Lahmeyer. *Application of three-dimensional interactive modeling in gravity and magnetics.* In_ Geophysics, Vol. 53, No. 8, pp. 1096-1108, 1988

[Mal92] J.L. Mallet. *GOCAD: a computer aided design program for geological applications.* In: A.K. Turner, Three-Dimensional Modeling with Geoscientific Information Systems, NATO ASI Series C, Vol. 354, Kluwer Academic Publishers, pp.123-141, 1992

[OD97og] Object Design Inc. *ObjectStore C++ API User Guide.* ObjectStore C++ release 5.0 documentation.

[OGI96] The OpenGIS™ Guide. Introduction to Interoperable Geoprocessing. Part I of the Open Geodata Interoperability Specification (OGIS). OGIS Project Technical Committee of the Open GIS Consortium Inc., Buehler K. and McKee L. (eds.), OGIS TC Document 96-001. http://ogis.org/guide, 1996

[OMG97c] Object Management Group. *CORBA 2.0/IIOP Specification*. OMG formal document 97-09-01. http://www.omg.org/corba/c2indx.htm

[OMG97s] Object Management Group. *CORBAservices: Common Object Services Specification.* OMG formal document 97-07-04. http://www.omg.org/corba/csindx.htm

[RaL97] J. Raper, D. Livingstone. *Integrating spatio-temporal coastal simulation with GIS.* In: Abstracts of the European Science Conference on space-time modelling of bounded natural domains: Virtual environments for the geosciences, Kerkrade, Netherlands, 1997

[RB+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *Object-Oriented Modeling and Design,* Prentice Hall, New Yersey, 500p., 1991

[ScW93] H.-J. Schek, A. Wolf. *From Extensible Databases to Interoperability between Multiple Databases and GIS Applications,* SSD'93, Signapore, LNCS No. 692, Springer, Heidelberg et al., pp. 207-238, 1993

[Sie88] A. Siehl. *Construction of Geological Maps based on Digital Spatial Models*. Geol. Jb., A104, Hannover, p.253-261, 1988