

## Integration of 3D Geoscientific Visualisation Tools with help of a Geo-Database Kernel<sup>1</sup>

Serge Shumilov and Martin Breunig  
Department of Computer Science III,  
University of Bonn, Germany  
e-mail: [shumilov@cs.uni-bonn.de](mailto:shumilov@cs.uni-bonn.de)

### Abstract

Visualisation and the handling of large data sets are key issues in the workflow of many applications in today's life and engineering sciences. However, from a software and data engineering point of view, these two important services are often located in two separated, i.e. independent and closed software systems. This means that the exchange of data and operations between software systems, i.e. the visualisation of database queries and the storage of visualised objects in the database management system, respectively, are not well integrated into many applications systems.

We present an open CORBA-based system architecture that connects two existing geoscientific software tools – the geological 3D modelling and visualisation tool *GOCAD*<sup>2</sup> and the geophysical 3D modelling tool *IGMAS*<sup>3</sup> – via a 3D Geo-Database kernel, *GeoToolKit*<sup>4</sup>. The common geo-scientific objects in the database can be accessed from *GOCAD* and *IGMAS*. The advantage of this method is that the 3D modelling tools can remotely access not only data but also the 3D advanced geometric database operations of *GeoToolKit*. Moreover, using the standard middleware platform we make the database and data also accessible for other applications. We report about our experience of the implementation with ObjectStore ODBMS and a CORBA/ODBMS adapter. Implementation aspects and open problems are discussed.

### 1 Motivation

The necessity of an interconnection between geoscientific modelling software and database systems is recognised and accepted since many years. Nevertheless, the integration of geoscientific modelling applications and databases, allowing a direct access from the modelling software onto the database system and vice versa, is not a standard within geoscientific applications.

Usually, not only one geoscientific application, but also varieties of applications are used for modelling, validation or visualisation of a specific area. Therefore, each of these applications interacts with each other by the exchange of data. The storage and management of all these data in separate files is a common method, though it is hazardous. At this, small scripts have to be written to convert a data set from one application to another. The representation of data

---

<sup>1</sup> This work is funded by the German Research Foundation (DFG) within the collaborative research centre SFB350 at Bonn University and the joint project "Interoperable geo-scientific information systems".

<sup>2</sup> *GOCAD* - geological 3D modelling tool. <http://www.ensg.u-nancy.fr/GOCAD/>

<sup>3</sup> Interactive Gravity and Magnetic Application System (*IGMAS*) - 3D gravity and magnetic modelling program [http://userpage.fu-berlin.de/~sschmidt/Sabine\\_IGMAS.html](http://userpage.fu-berlin.de/~sschmidt/Sabine_IGMAS.html)

<sup>4</sup> *GeoToolKit* - geometrical 3D kernel of the spatial ODBMS *GeoStore*. <http://www.geo.informatik.uni-bonn.de/software/GeoToolKit>

into different formats leads to a redundant storage of data. It is (a) memory intensive and (b - and more important) might lead to working on the "wrong" data.

The use of DBMS as storage for the common spatial data has several advantages for the support of geoscientific modelling, e.g. an automatic *integrity control*. Spatial and temporal data can be tested against their semantic plausibility. The integrity checks can be executed during the reading of the data into the database or during the execution of database queries. Last, but not least, *security mechanisms* guarantee data security.

The paper is organised as follows. In chapter 2 we briefly introduce an object-oriented 3D-4D spatial database *GeoStore* and the application design technology evolved on top of *GeoToolKit*. In chapter 3 we introduce extensions to *GeoToolKit*'s data model, which allow storing common data from two geomodelling tools *GOCAD* and *IGMAS*. A CORBA-based system architecture that integrates both systems with the database into one distributed 3D GIS is presented in chapter 4. This chapter also discusses existing CORBA/ODBMS integration techniques for providing distributed support for existing databases. A short overview of our approach to CORBA/ODBMS integration based on the eXtensible Database Adapter (XDA) is presented. Chapter 5 presents the evaluation of the integrated system and outlines some possible future developments and the last chapter summarises the contributions of this article.

## 2 Case Study

In our case study the coupling between 3D-modelling tools through an object-oriented geo-database kernel system was implemented to support a new method for the generation of geological maps. The geoscientific goal of this research was the development and implementation of a method, which allows the construction of plausible geological maps. At this, the geological map is generated by the cut of a 3D stratigraphical model with a digital elevation model (DEM).

The 3D model serves also as input for subsequent investigations, e.g. numerical simulations on kinematic or dynamic models, by which an additional control of the generated 3D model is possible. In the study presented here, the geological 3D model is validated by an iterative exchange between geological and geophysical modelling tools to ensure geoscientific consistency. Here, besides *GOCAD* – a 3D geological modelling tool, an additional, geophysical application *IGMAS* is used to provide further constraints for an iterative revision of the 3D model and thus a succeeding modification of the geological map. For this reason, the 3D stratigraphic model, which is used as base model for the subsequent 3D gravity modelling using *IGMAS* and served as input for subsequent steps of validation using gravity and susceptibility data. However, the description of the iterative validation of the 3D model is not subject of this paper. It can be found in (Breunig et al. 1999).

## 3 GeoStore - a common object-oriented spatial DBMS

The integration of both (*GOCAD* and *IGMAS*) tools through a common Geo-Database enables the consistent handling of both, data and 3D models (and parts of it) during the geoscientific modelling process. The advantages, which arise by the use of the common database, are manifold, concerning not only the management of data and models but the management of multi-user access as well. At this, the most important aspects are (a) a central management of all data, (b) a simplified incorporation of new data, resulting in an implicit actualisation of the data, instantly visible to all participating editors and (c) update support of queries of data and models.

As a basis for development of a *common* data store it was reasonable to use the existing geodatabase system *GeoStore* developed within the collaborative research centre SFB350 at the University of Bonn (Bode et al. 1994). The intention of *GeoStore* was to supply geoscientists with a tool, which would provide the consistent storage and efficient access for data involved in all stages of the interactive 3D modelling process (Siehl 1993). The first version of *GeoStore* was implemented on top of the RDBMS Oracle. However, because of the complex structure of spatial data with multiple interlinks between objects, the object-oriented data model turned out to be more suitable for the development of a database. Consequently, *GeoStore* was completely re-implemented with the ODBMS ObjectStore (ODI 1997).

The first step towards interoperability between the geological and the geophysical 3D modelling systems (*GOCAD* and *IGMAS*) was the development of a common object model. The most important requirement here was that geological and geophysical representations for the same entities might differ both in thematic attributes and in representations for geometric and topological data. The diversity of representations arises from the distinct purposes they serve for. Thus, in *IGMAS* data structures are optimised for the efficient mathematical computations while in *GOCAD* classes are oriented toward an efficient geometric editing and visualisation.

### 3.1 *GeoToolkit* – geometric database kernel

Therefore, to meet the requirements of geoscientists occurring during the construction of 3D models, a common database model should be general enough to allow the integration of data types from both applications. To provide a non-redundant maintenance and cooperative utilisation of data by different applications developed within the interdisciplinary geoscientific research centre at Bonn University, the geometric kernel of *GeoStore*'s class hierarchy is separated from the other functionality in an independent library - *GeoToolkit* (Balovnev et al. 1997).

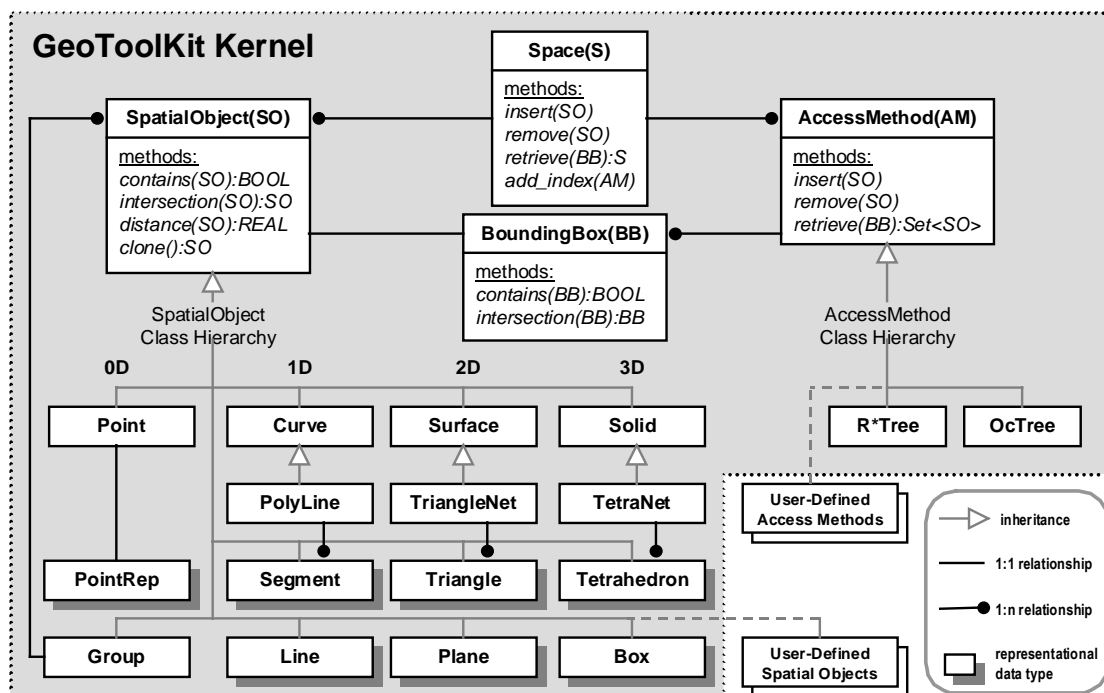


Figure 1: *GeoToolkit's object model*

*GeoToolKit* is a C++ class library that was designed with special respect to generality and extensibility. It is intended to facilitate the development of 3D/4D geo-scientific applications in domains that are not covered by general-purpose geo-information systems (GIS). *GeoToolKit* provides to application developers a set of geo-oriented building blocks involving ODBMS-based spatial data maintenance, visualisation, graphical interfaces and communication that can be assembled into a ready-to-use application by geo-scientists with little programming experience. Currently, *GeoToolKit* offers classes for the representation and manipulation of simple (point, segment, triangle, tetrahedron) and complex (curve, surface, solid) 3D spatial objects (Figure 1). The *GeoToolKit* class hierarchy is complete: any spatial object can be modelled either directly by one of the built-in spatial classes or as a composition of these classes within a group - a heterogeneous collection of spatial objects, which are further treated as a single object. However, being general enough, *GeoToolKit* is more than just a set of empty interface specifications, providing a class hierarchy for the representation of various geometric bodies. The library addresses primarily the efficient storage and retrieval of 3D-spatial objects within a database directly motivated by the needs of the interactive 3D modelling. To achieve this, *GeoToolKit* is tightly coupled with the object-oriented DBMS ObjectStore; e.g. we intensively exploited its low-level clustering control facilities in the development of spatial indexes.

### 3.2 Integrated Data Model

The developed geological-geophysical data model is based upon *GeoToolKit*'s class hierarchy. Figure 2 presents a part of the data model that inherited from two basic *GeoToolKit*'s classes: *Surface* and *Volume*. Their relationships are shown in the OMT-like notation (Rumbaugh et al. 1991).

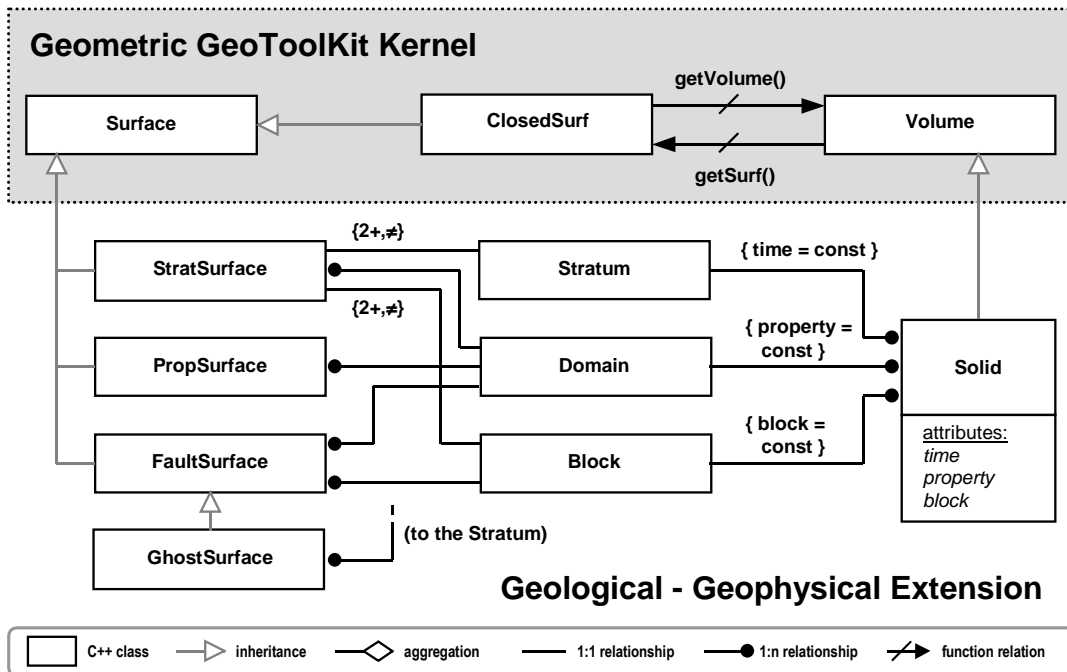


Figure 2: Integrated data model.

Any newly defined data type customised according to the conventions of *GeoToolKit* can be included in the class library for further re-utilisation. New geo-scientific classes define some additional thematic attributes (e.g. lithology, tectonics and density) and advanced geological

operations like the interpolation of values from a point set to the surface of a triangle network. A new type can inherit representation and functionality from one of the embedded classes redefining its functions to get, for example, a more efficient implementation. All geo-scientific classes can be subdivided into three sections: a) surface based spatial compound objects: *StratSurface*, *PropSurface*, *FaultSurface*; b) volume based spatial compound objects: *Stratum*, *Domain*, *Block* and one volume based spatial complex: *Solid* (Figure 2). *Solid* is a maximal volume that has *constant* thematic attributes. *Solid*'s objects are used as basic building blocks for compound objects. For example, groups of solids with the same lithology attributes build a *Stratum*. This solid representation is more preferable for geological modelling, because it allows an efficient derivation of stratigraphical values for arbitrary locations in space.

In *GOCAD*, a solid (referred to as *Tsolid*) can be modelled as a specialised tetrahedron container class, which is derived from a general simplicial complex class. Alternatively, a stratum can be modelled as a composition of bounding surfaces: faults and an upper and lower stratigraphic boundaries. *IGMAS*, on the contrary, exploits the bounding surface representation under assumption that bodies have no holes. The advantage of *GeoToolKit* for integration purposes here is that it supports flexible switches between different spatial representations. For example, the tetrahedron network representation for a solid in *GOCAD* can be easily converted (using *GeoToolKit*'s embedded methods) into a bounded surface representation, utilised in *IGMAS*, and vice versa.

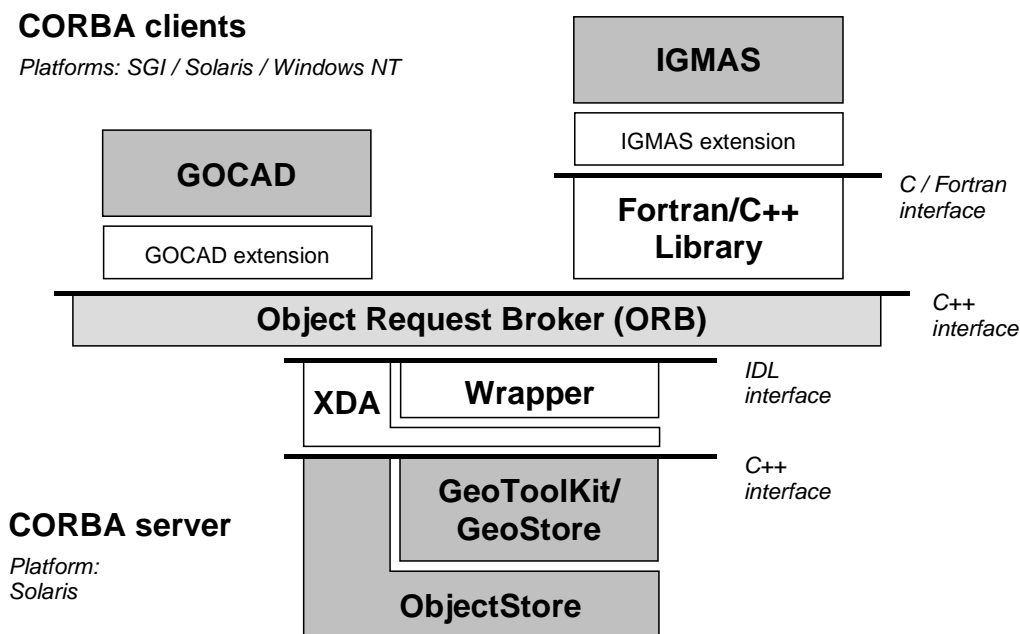
#### 4 CORBA Based Distributed Architecture

During the development of the remote access to *GeoStore* it became evident that low-level facilities provided within the UNIX operating system (sockets, Remote Procedure Call) and direct ObjectStore database-level client-server communication have many drawbacks (Balovnev et al. 1998). To avoid these hardships we have gone over to a standard distributed computing platform - *Common Object Request Broker Architecture* (CORBA) (OMG 1999). Using the standard architecture we can expect that any other CORBA-conform application can access *GeoToolKit*-based data stores. A big 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.

The general schema of the CORBA based integrated architecture is presented in Figure 3. As the CORBA implementation in our prototype Orbix from IONA Inc. (IONA 1998) was used. It provides a transparent network access between different computation platforms such as Sun Solaris, SGI IRIX and Windows NT. The *GeoStore* database system was made available to the CORBA environment with help of a special wrapper, which covers ObjectStore's native C++ programming interface of the database with a new CORBA compatible interface using an Interface Definition Language (IDL). The wrapper was developed on top of an *eXtensible Database Adapter (XDA)* – our prototype of a CORBA/ObjectStore object adapter, providing wrapper developers with the full power of ObjectStore to define, manipulate, and share persistent data in CORBA (Shumilov & Cremers 2000).

Covered by the wrapper, *GeoToolKit/GeoStore* can be regarded as a distributed 3D/4D geo-database available for remote CORBA-compatible clients. In our case, the clients are *GOCAD* and *IGMAS*. *GOCAD* is a C++ application. That is why it can directly use Orbix's client IDL/C++ mapping for database access. The second – *IGMAS*, is a Fortran application that can not directly access the CORBA interface, since Orbix has no IDL/Fortran compiler. For that

purpose, an additional Fortran/C++ library was developed. Extensions to the graphical user interface for both programs enable the manipulation with remote data stores. Objects can be loaded from and stored into *GeoStore*'s data stores. For example, a stratigraphic boundary surface that before was constructed in the geological 3D-modelling tool *GOCAD* can be stored in the database and loaded again from *IGMAS*.



**Figure 3:** General schema of CORBA based integrated architecture.

#### 4.1 General approaches to the CORBA/ODBMS integration

A primary strength of object-oriented databases lies in their ability to model complex objects and inter-relationships among them. In contrast to database systems, CORBA provides a flexible transparent object-oriented distribution model with a larger-scale set of services. Integration of both technologies permits the encapsulation of the powerful database facilities within the heterogeneous CORBA environment.

CORBA does not support object persistence directly, but allows the integration with different database systems. Usually it is a construction of a middle-tier layer between the CORBA clients and the database that wraps the original database interface and provides an equivalent CORBA compatible interface. Traditionally, the layer's implementation is named a *wrapper*. The actual CORBA specification (OMG 1999) defines two standard ways that help to reduce costs of the wrapper's development: *Persistent Object Service* and *Object Database Adapter*.

The *Persistent Object Service (POS)* is one of the standard Object Services specified in the OMG document (OMG 1998). Unfortunately, the standard is very general and complex. Two examples are the vague semantics of the operations and the weak specification of how POS interact with the database and other Object Services. Consequently, POS does not yet have any complete implementation and therefore we will not consider it as a sound approach. The second version of this service – the *Persistent State Service* is now under development.

A service like POS typically does not require extensions to the ORB and its standard components: it is just treated like a regular application. Conversely, an *Object Database*

*Adapter (ODA)* approach provides a tight integration between the Object Request Broker (ORB) and the database system. At first, the idea of the special object adapter for persistent objects was pursued in the ODMG standard (ODMG 1994) and later implemented in some projects (Reverbel & Maccabe 1997). The ODA is an object adapter for objects stored in a database. It does not completely replace the standard object adapter, but represents its extension that depends on a particular database system. ODA is responsible for the managing of the persistent state of implementation objects and. Its objective is the simplification of their programming. It should also deal with other typical database features, e.g. locks and transactions, since persistent objects are generally shared.

#### **4.2 Proposed approach - an eXtensible Database Adapter**

Using our own experience gained during the evaluation of commercially available ODA's, we developed an alternative object adapter - the *eXtensible Database Adapter (XDA)*. In our approach we tried to focus on the most critical issues of the CORBA/ODBMS integration that we missed in commercial adapters. Thus we paid more attention to support distribution for existing databases. In this paper, we will only outline the most significant case study aspects of the XDA functionality. A detailed description of XDA can be found in (Shumilov & Cremers 2000).

***Soft, non-intrusive integration:*** The main advantage of XDA is the possibility of a soft, non-intrusive CORBA-integration. It is very important for existing databases, since database schema evolution may be extremely time-consuming and any modifications in the schema and in applications are not possible. To prevent them, XDA adds an intermediate layer of *transient* intermediate communication components that act as usual CORBA servers as well as database clients. In other words, they *mediate* between CORBA client and database objects converting data parameters and delegating all function calls in both directions. Because of this basic function, we will name such components *mediators* (Wiederhold 1999). However, the price of this improvement is the responsibility of XDA to control the lifetime of mediators – their creation, deletion and the binding to the corresponding database objects.

***Persistence of object references:*** CORBA clients work with remote server objects through *Interoperable Object References (IORs)* that identify remote objects in the CORBA environment. The difference between object references to transient and persistent objects is that the latter are valid during substantially longer time, e.g. a reference could be stored or exchanged by the clients. To provide persistence of the references, XDA modifies IOR saving additional information in the marker. This means that a client, having an object reference can use it at any time without warning, even if the mediator has been deactivated or the server system has been restarted. With the persistence of object references, it makes perfect sense for the client to store an object reference for later use or send it to other clients. For example, references to the persistent CORBA objects implemented by a server X can be stored by a server Y (a client of server X) and vice versa, thereby enabling the construction of ORB-connected multidatabases.

***Transaction management:*** In the databases an access to persistent data is controlled with the help of transactions. Before a program can access persistent data, it must start a transaction. By default, transaction's boundaries and types are implicitly controlled by the XDA. It guarantees that the access to persistent data will always take place within a transaction and performs a "conservative" policy to their control. The adapter always tries to prevent the start of a new transaction, and if possible, it uses an opened transaction. It also makes a difference

between "read\_only" and "update" transactions, preferring the first to the second. In the worst case, if a current transaction can not be reused, it will be closed and a new one will be started.

If a client wants to manipulate boundaries of transactions, it can use the corresponding methods from the XDA's IDL interface. They allow the client to begin, commit or abort a transaction explicitly whenever it is necessary. The XDA also allows the client to nest transactions and it supports all transactional modes provided by ObjectStore.

The main advantage of the *implicit* automatic mode is that it allows CORBA clients to work with persistent objects in the same way that they did with usual CORBA objects. This mode is quite usable for "pure" CORBA clients that do not know that they work with persistent objects and do not care about transactions at all. Usually after the client's explicit start of a new transaction, the XDA switches off the automatic transaction control delegating this responsibility to the client. However, the client can explicitly switch it on later. For example, it might be useful in the situation where all nested operation calls should be done within separate sub-transactions. It may be inefficient to explicitly start and to end transactions so frequently.

## 5 Evaluation

The implementation of the *GeoStore* object model as an integrated geological-geophysical extension of *GeoToolKit* proved the advantages of the *GeoToolKit*-based development. A significant part of geometric relationships between geological objects in *GeoStore* was inherited from the *GeoToolKit* classes. This results in a considerable reduction of code that not only accelerated the development process, but also increased the reliability of applications preserving from inevitable errors during the new development. The developers can focus on the application semantics instead of such "creative" tasks like optimal assembling of spatial objects from multiple tables or the implementation of routine geometric algorithms. Since application specific classes no longer contain geometry-related components, the classes become significantly shorter and, consequently, more understandable for external users. Developing all applications on the common *GeoToolKit* basis, we make a significant contribution in the non-redundant and consistent maintenance of shared data. Data designed and stored within a particular application become available for other *GeoToolKit*-compliant applications as well.

One of the most attractive features of a DBMS for geologists, however, is the automatic *integrity control*. Common underlying data structures provide a necessary basis for the successful integration and interoperation of geo-scientific systems built on top of *GeoToolKit*. Spatial and temporal data can be tested against their geological plausibility. The integrity checks can also be explicitly performed during the reading of the data into the database or during the execution of database queries. With regard to the example from Southern Lower Saxony presented here, the integration of geological and geophysical modelling tool leads towards a better plausibility and an improved validation of the 3D model. A free data exchange via a common database have given geo-scientists an opportunity to perform a cooperative work under one object model, e.g. the adjustment of geophysical density and geological stratigraphic parameters (Breunig et al. 1999). This work resulted in an increased consistency of the initial data taken from the "Geotectonic Atlas of North-West-Germany" and were included in the new version of this atlas (Kockel et al. 1999).

The integration of *GeoStore* with CORBA has made *GeoStore*-based databases concurrently accessible for multiple clients from different platforms and programming environments. The construction of *GeoStore*'s wrapper shows that the use of XDA has substantially reduced the wrapper's development process and improves performance of the integrated CORBA/ODBMS system. The transience of mediators not only saved database space, but also allowed us to make the integration without any changes in the original database schema and without disturbing of existing local applications. It also separates the original database and CORBA components provided by XDA allowing concurrently work with the database through both – CORBA and native C++ database communication interfaces. Moreover, when employing this method, any already existing data store providing a *GeoToolKit* compatible schema can always be used through the CORBA-interface.

This integration also allowed us to speed up all operations, which need to process large amounts of data by performing them locally at the server site. This is especially interesting for the web browser access to databases. With the integration of the Java language into the ORB environment, Java applets can interact with the persistent CORBA objects through domain-specific interfaces, without any knowledge of how the objects are actually stored. In the future, we have to consider the application of XML for the definition of the data types transmitted, as XML is a standard and will be employed to enable data transfer between different applications across heterogeneous platforms.

Referential integrity of the data shared by local and CORBA-conformant applications is guaranteed by the use of the native ObjectStore transaction control mechanism. However, the long duration of geo-transactions would require long locks for those database objects that have been requested by the user query. That is why an automatic transaction control by XDA was used. Besides standard tasks, the mediating tier residing between both systems must also perform some additional sophisticated functions aimed to improve performance of the distributed access. The evaluation of the XDA prototype shows that management of mediators and transactions are the most important adapter's tasks that have great influence on the scalability and performance of the integrated system. This topic will remain one of the most important fields, where more and more refined techniques should be applied. Future developments of the CORBA/ODBMS integration process will be certainly influenced by the modern rapidly evolving technologies such as XML and Java.

## 6 Outlook

The coupling between *GOCAD* and *IGMAS*, 3D-modelling tools through *GeoToolKit*, a common CORBA-integrated object-oriented 3D-geodatabase-kernel system has been proposed. Hitherto we have implemented a prototype of a 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 an opportunity for geo-scientists 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 optimised in a wide area network.

## 7 Acknowledgements

The authors would like to thank the German Research Foundation (DFG) for the financial support within the collaborative research centre SFB350 at Bonn University and the joint project "Inter-Operable Geo-scientific Information Systems". This work would also have been impossible without the excellent cooperation with the groups of Agemar Siehl (Bonn University) and H.-J. Götze (Free University Berlin). We like also to thank our partners BEB Erdgas und Erdöl GmbH, Mobil Oil AG, Preussag, Winthershall AG and NLfB for providing data.

## 8 References

- Balovnev, O., Breunig, M., Cremers, M., "From *GeoStore* to *GeoToolKit*: The second step", In: *proceedings of the 5<sup>th</sup> Intern. Symposium on Spatial Databases*, LNCS 1262, Springer, Berlin, pp. 223-237, 1997.
- Balovnev O., Bergmann A., Breunig M., Cremers A.B. & Shumilov S., "A CORBA--based approach to Data and Systems Integration for 3D Geoscientific Applications". In: *Proceedings of the 8th Intern. Symposium on Spatial Data Handling (SDH'98)*, Vancouver, Canada, pp. 396-407, 1998.
- Bode, T., Breunig, M., Cremers, A.B., "First Experiences with *GeoStore*, an Information System for Geologically Defined Geometries". In: *Proceedings of the Intern. Workshop on Advanced Research in Geographic Information Systems*, Ascona, Switzerland, LNCS 884, Springer, Heidelberg, pp. 35-44, 1994.
- Breunig, M., Cremers, A.B., Götze, H.-J., Schmidt, S., Seidemann, R., Shumilov, S., Siehl, A., "First Steps Towards an Interoperable GIS - An Example From Southern Lower Saxony", In: *Physics and Chemistry of the Earth*, Vol. 24, No. 3, S. 179-190, 1999.
- IONA Technologies Ltd., *Orbix Programmer's Guide*, Version 2.3, October 1998.
- Kockel, F. et al., *Geotektonischer Atlas von Nordwest-Deutschland und dem deutschen Nordsee-Sektor - Digital*, Hannover, 1999. <http://www.bgr.de/>
- Object Management Group, *CORBA services: Common Object Services Specification*, 1998.
- Object Management Group, *CORBA/IIOP 2.3.1 Specification*, 1999.
- Object Database Management Group, *The Object Database Standard: ODMG-93*, R. Cattell (ed.), Morgan Kaufmann, 1994.
- Object Design Inc., *ObjectStore C++ API User Guide*, Release 5.0, Object Design Inc., 1997.
- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W., *Object Oriented Modelling and Design*, Prentice Hall, 1991.
- Reverbel, F., Maccabe, A., "Making CORBA Objects Persistent: the Object Database Adapter Approach", In: *Proc. of the 3rd USENIX Conference on Object-Oriented Technologies and Systems (COOTS'97)*, Portland, Oregon, pp. 55-65, June 1997.
- Siehl, A., "Interaktive geometrische Modellierung geologischer Flächen und Körper", In: *Die Geowissenschaften*, Berlin, 11. Jahrg., Heft 10-11, pp. 342-346, 1993.
- Shumilov, S., Cremers, A.B., "eXtensible Database Adapter - a framework for CORBA/ODBMS integration", In: *Proceedings of the 2<sup>nd</sup> International Workshop on Computer Science and Information Technologies (CSIT'00)*, Ufa, Russia, 2000.
- Wiederhold, G., "Mediation to Deal with Heterogeneous Data Sources", In: A. Vckovski, K.E. Brassel, H.-J. Schek (Hrsg.): *Interoperating Geographic Information Systems, Proc. of the 2<sup>nd</sup> Int. Conf.*, Zurich, Switzerland, Lecture Notes in Computer Science, no. 1580, Springer, pp. 1-16, 1999.