Netherlands Journal of Geosciences / Geologie en Mijnbouw 81 (2): 251-256 (2002)

Spatial and Temporal Database Support for Geologists – An Example from the Lower Rhine Basin

M. Breunig[1], O. Balovnev[2], A.B. Cremers[2] & S. Shumilov[2]

[1] Institute of Environmental Sciences, University of Vechta, P.O. Box 1553, 49364 Vechta, Germany; corresponding author; e-mail: mbreunig@iuw.uni-vechta.de
[2] Institute of Computer Science III, University of Bonn, Roemerstr. 164, 53117 Bonn, Germany; e-mail: (abc,oleg,shumilov)@cs.uni-bonn.de

Manuscript received: December 2000; accepted: January 2002

Abstract

Within the collaborative research centre SFB350 at the University of Bonn a component software called *GeoToolKit* has been developed, which is intended to facilitate the design and implementation of 3D/4D geological applications. It provides a range of geo-oriented software building blocks for application developers involving database management system based spatial and temporal data maintenance, support for efficient spatial and temporal retrieval, communication, visualization and graphical interfaces which the user could assemble in a ready-to-use application. As such, *GeoToolKit* is not a GIS-in-a-box package – rather it is a library of C++ classes that allows the incorporation of spatial functionality within an application under development. It is primarily oriented towards software engineers with the C++ experience involved in the development of special-purpose geological applications, which can hardly be modelled within standard GISs. We present applications using GeoToolKit, which have been developed to support the geological reconstruction of the Lower Rhine Basin.

Keywords: spatio-temporal database, geotoolkit, 3D/4D GIS, geological information system

Introduction

GeoToolKit (Balovnev et al., 1997) evolved on the experience we gained within the collaborative research centre during the development of diverse geological applications (Jentzsch & Siehl, 2002; Thomsen & Siehl, 2002). *GeoToolKit's* architecture and functionality was initially inspired by the requirements of *GeoStore* (Bode et al., 1994) – an information system designed to manage 3D geologically defined geometries of the Lower Rhine Basin. These requirements especially consider the support of the interactive geological modelling with spatial and temporal database queries. At the same time we tried to make a toolkit general enough to be used in the development of a possibly wide range of applications. However, it was obvious for us that a toolkit should be more than just a set of empty interface specifications. *GeoToolKit* should provide at least one complete implementation for all object types and functions specified in the interface.

Methods

Modelling geological structures and history, *GeoStore* always starts with a careful geometric analysis of the present assembly of geological surfaces, bodies and property distributions, this being the key to reveal the nature and interaction of earlier processes. Computer-aided constructions of consistent geometric models are known in geology *as interactive 3D/4D modelling* (Alms et al., 1996). The models are further tested by backward restoration and balancing. The final model is used for the specification of boundary conditions for 3D process models, or for the production and consistent updates of geological maps.

The starting point for the interactive geological 3D/4D-modeling is the digitisation of geological *sections* gained from opencast lignite mining. A stratum is modelled as a list of layered bodies, which are usually specified by their bounding surfaces. A fault is modelled as a single surface. Within a section, strata and faults are represented as point sets grouped in stratigraphic and fault lines. Each point in a section containing 3D-coordinates is complemented by geologically specific data such as stratigraphy. The second step in the interactive 3D modeling is the generation of *triangulated surfaces* spread between sections. Finally, the last step is the transition from stratigraphical bounding surfaces to *volumes*.

Results and Analysis

GeoStore

The intention of *GeoStore* has been to supply geologists with a tool which should provide a consistent storage and efficient access for the data involved in all stages of the interactive 3D-modeling using modern database technology. We began the development of *GeoStore* with the elaboration of an object data model.

There are three basic classes (see Fig. 1): *Section*, *Stratum* and *Fault* with extensions used as entry point to the database. A stratum in a section (*StratInSection*) contains an ordered list of stratigraphical lines. A stratigraphical line (*StratLine*) is a specialization of a geometric line, extended for efficient computations with the additional topological information, namely a list of stratigraphical lines above the given (*hanging wall*) and the bounding faults (*left fault* and *right fault*).

Using this object data model, *GeoStore* supports two basic types of spatial queries:

Queries on strata and faults within a section. The user, for example, may select all strata with a certain hanging stratum in a given depth interval located between specified faults. This type of query uses primarily the topological relationships explicitly stored in the database.

Queries on triangulated stratigraphic and fault surfaces. This type of query primarily deals with geometric 3D-operations like horizontal or vertical slices through stratigraphic and fault surface assemblies, an intersection between surfaces or clipping of a part of a surface within a bounding box.

Redesigning GeoStore on top of GeoToolKit

The main disadvantage of GeoStore has been that its geometric classes could only be used for the special application they had been designed for. Therefore we decided to separate the geometric classes from the geological classes of the applications. The geometric class library should be realized in our GeoToolKit, whereas the application-specific classes should be implemented in the redesigned version of GeoStore.

While redesigning *GeoStore* on top of *GeoToolKit* (see Fig. 2), the main design principle we followed was to inherit from the geometric kernel as much functionality as possible. Fig.1 shows the class diagram of the new GeoStore system on top of GeoToolKit in an object modelling technique-like notation (Rumbaugh et al., 1991). Following this approach, a user-defined class automatically inherits the whole geometric functionality of its parent. For every geological entity we found the geometric 'pattern', which served as its ancestor in the class hierarchy. For example, a geological fault is represented now as a special-

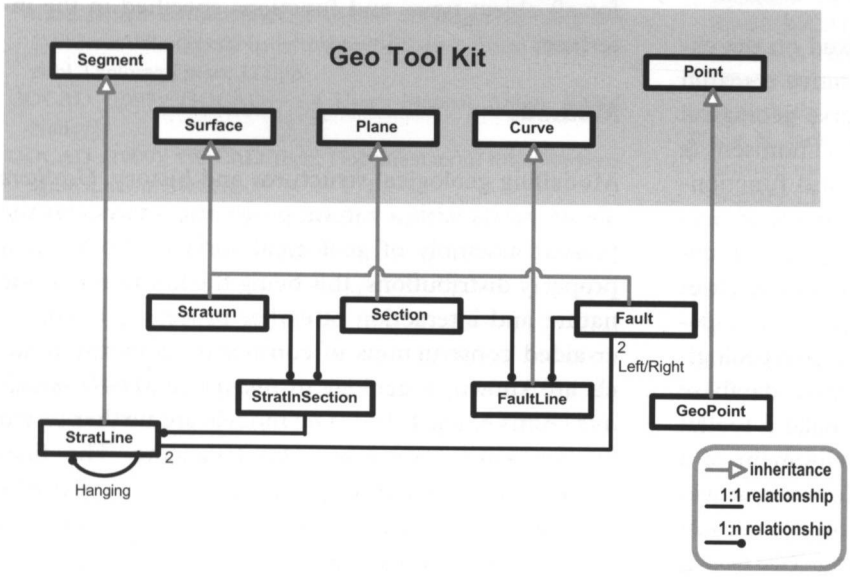

Fig. 1. GeoStore's class hierarchy.

https://doi.org/10.1017/S0016774600022502 Published online by Cambridge University Press

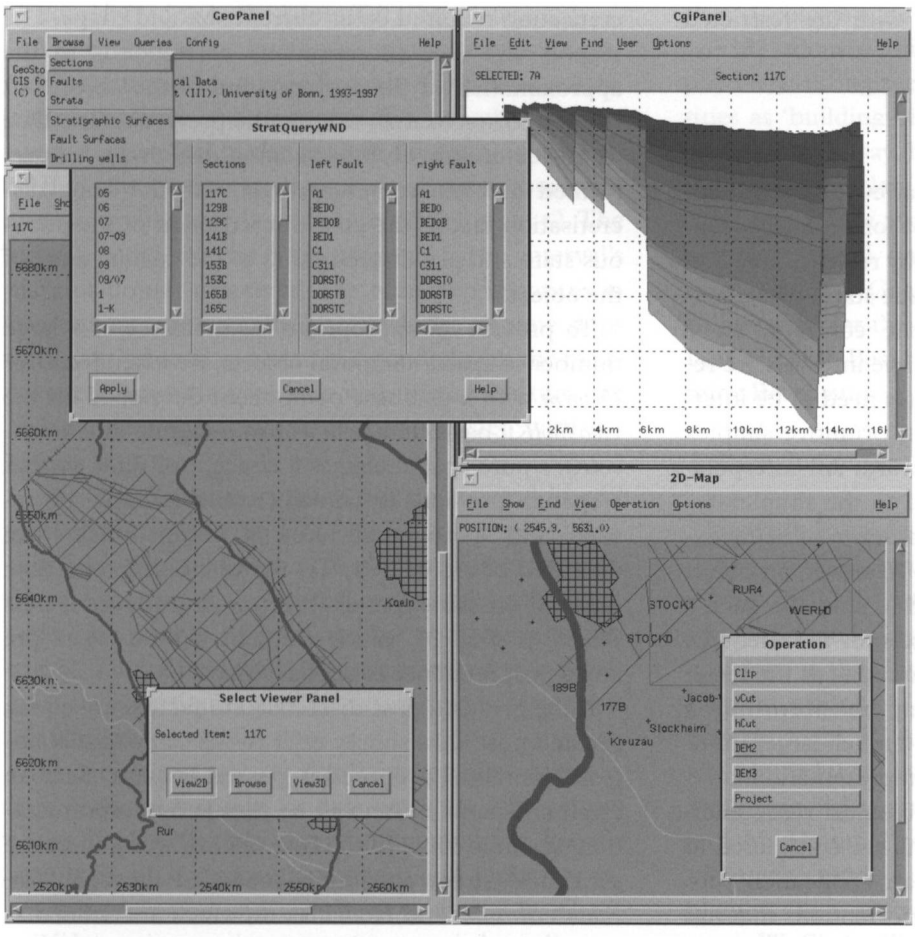

Fig. 2. Sections and geological faults of the Lower Rhine Basin visualised by GeoStore.

ization of the geometric entity *Solid*, which was extended with geology-specific features (e.g. stratigraphy) and relationships (e.g. the *Hanging* stratum). By using the inheritance every geological entity can be treated as a geometric object. *GeoToolKit*'s geometric kernel was able to operate directly on the collections of geological objects without intermediate transformations. In the case of the incorporation we would have to create a temporary space and to copy the geometric ingredients of geological objects – a task which may be too expensive in the database context.

A geological solid may have two alternative representations: a bounding surface approximated in turn by a triangle network and a tetrahedron network. The first one is widely used in computer graphics. However, we chose the alternative representation of solids, because there is one decisive advantage for geological applications: any internal point not only automatically inherits values that are characteristic to the whole solid. But it also preserves them during diverse geometric manipulations and transformations.

However, it was not easy to find a geometric pattern for such a sophisticated object like a geological section. There were conflicts between its representation as a set of plane surfaces or as a combination of curves. Therefore we implemented the *Section* class as a composite object containing both components. Since *Section* inherits directly from the abstract *SpatialObject* class, it provides both the representation and the implementation for all functions specified in the *SpatialObject* class. Practically, the re-implementation resulted in delegating the functionality to elementary geometric components.

GeoStore is characterized by its tight coupling between database and visualization. Due to this, the user is able to switch at any time between the object browser and one of the internal 2D/3D-graphical viewers (Fellner, 1996), which are implemented as specialization of the *GeoToolKit* embedded viewers. In addition to a default model, the user can assign their own interpretation to a service object. For example, on a 2D-map a line segment can be used to specify a bounded vertical plane, which proves to be a very convenient method to specify vertical slices of subsurface structures, i.e. geological as cross-sections.

Topological relationships (e.g. neighbourhood) are used to provide very useful 'guidelines' for modelling, not available earlier in the 'pure' geometric world. In order to benefit from this additional information, we simply re-implemented virtual functions. An overridden function either substitutes the geometric function completely. Or it performs a kind of pre-selection for

the geometric function calling it with the restricted subset of spatial objects.

Spatio-Temporal Data Browser

The ability to follow a topological evolution of geological entities is of special interest for geologists. Geological entities are characterized by relative large and irregular time intervals between time states where sufficient data are available. On the contrary, for a smooth animation, small regular time intervals are required. Therefore missed time states need to be interpolated. An interpolation between primitive simplex objects (simplexes) is straightforward. An interpolation of complexes can be reduced to the simplex-to-simplex interpolation only if the number of complexes is the same in both time states. However, an object may change with time its size and/or shape in such a way that it will need more simplexes for the adequate representation than before. For example, as the result of deformations a flat surface may transform into a spherical surface, which will need a much larger number of triangles for the qualitative representation.

The special geometric and topological representations for moving geological objects are considered in *GeoDeform* (Alms et al., 1998), an application program for calculating geological deformations that has been developed at the Geological Institute at Bonn University. For the visualization of spatial objects changing in time, *GeoDeform* uses a model proposed in the graphical library GRAPE (Polthier & Rumpf, 1995). According to this model, each time state contains two representations of the same object with a different number of simplexes. The first representation (post-discretisation) corresponds to the approximation of the current state of the object with the discretisation required by its current size and shape. The second one (pre-discretisation) corresponds to the approximation of the current state of the object but with the discretisation used in the previous state. Due to this extension, an interpolation can always be performed between representations with the same discretisation factor: the post-discretisation of the previous state and pre-discretisation of the current state of the object.

To provide an appropriate maintenance of a large number of spatio-temporal objects, we extended *GeoDeform* with a database component developed on the *GeoToolKit* basis. The task was to integrate into *GeoToolKit*'s pure spatial classes a concept of time, so that a spatial functionality already available could be reused and a maximal level of compatibility with GRAPE was provided. To represent different time states of the same spatial object, we introduced a class *TimeStep* (Fig. 3), which contains a time tag, a pre and a post reference to spatial objects.

If the pre- and post-discretisation factors are equal, pre and post links simply refer to the same spatial object. The model proposed is general enough, since an arbitrary spatial object can be chosen as a representation of a time state. In the case of *GeoDeform* there are geological strata and faults modelled through *GeoStore*'s *Stratum* and *Fault* classes, which are defined as a specialization of *GeoToolKit*'s *Surface* class.

A sequence of *TimeStep* instances characterizing different states of the same spatial object is gathered into a spatio-temporal object (class *TimeSequence*). Being a specialization of the abstract class *SpatialObject*, a sequence can be treated in the same way as any other spatial object, i.e., it can be inserted into a space and participate in all geometric operations. The spatial functionality is delegated by default to the spatial

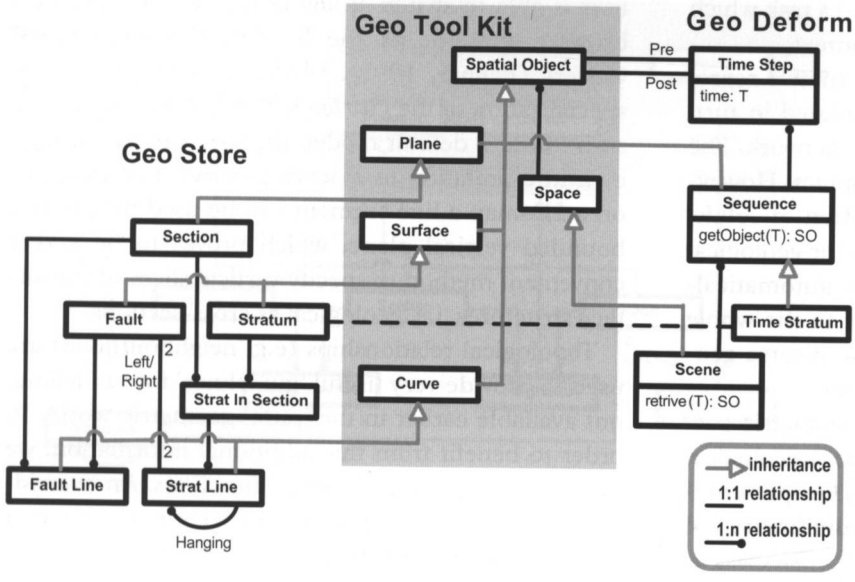

Fig. 3. Spatio-temporal extensions for *GeoToolKit*.

https://doi.org/10.1017/S0016774600022502 Published online by Cambridge University Press

object referred to in the latest *TimeStep* instance.

A selection for the interpolation differs from a common selection with a specified key. If there is no object in the time sequence that hits exactly the time stamp t (specified in the retrieve function) instead of NULL, the query returns a pair of neighbour time steps with time tags t1 and t2, so that $t1 < t < t2$. The same is valid for the time interval: if the interval's margins do not exactly hit the time step instances in the sequence, the resulting set includes all time steps fitting the interval completely extended with the nearest ancestor of the time step with the lowest time tag and the nearest successor of the time step with the highest time tag.

Any *TimeSequence* instance can be inserted in and spatially retrieved from *GeoToolKit*'s space as any other spatial object. However, to perform a temporal retrieval we introduced a special container class (*Scene*), which is capable of both spatial and temporal retrieval. Though the data model presented was developed for a particular application, it turned out to satisfy various geo-applications.

As mentioned above, *GeoStore* benefits considerably from the fact that its data types are built on top of the *GeoToolKit* class library. All application-specific data such as stratigraphic surfaces and faults are retrieved, processed and visualized primarily by *GeoToolKit*'s runtime environment (geometric kernel).

Discussion and Conclusions

The partial re-design of *GeoStore* as well as the implementation of new applications on top of *GeoToolKit* proved the advantages of the *GeoToolKit*-based development. The *GeoStore* classes contain only geology-specific data members and methods. A significant part of geometric relationships between geological objects can be inherited from *GeoToolKit* classes. This results in considerable reduction of code that not only accelerates the development process but also increases the reliability of applications since the re-use of already approved methods preserves the developer from inevitable errors during the new development.

Developing applications on the common *GeoToolKit* basis leads to a significant contribution in the non-redundant and consistent maintenance of shared data. Data designed and stored within a particular application become also available for other *GeoToolKit*-compliant applications. Moreover, already existing geodata stores can be connected to *GeoToolKit*-compliant data sources.

Apart from this, the object-modelling technique turned out to be an appropriate environment for the communication between geo- and computer scientists. Geologists not experienced in software design quickly adopted this technique and successfully applied it, utilizing the pre-defined set of *GeoToolKit* entities as 'building blocks'. The object data model designed in this way, has been mapped one-to-one into the *GeoToolKit* class library.

In our future work, we intend to support more complex spatial and temporal database queries in GeoStore and GeoToolKit. The new developments will be used for the efficient retrieval of data from the Lower Rhine Basin in general and from the open pit mine Bergheim.

Acknowledgements

This work is funded by the German Research Foundation (DFG) within the Collaborative Research Centre SFB350 'Interactions between and Modelling of Continental Geosystems' at Bonn University and was done in the close cooperation with the Geological Institute of Bonn University. We particularly acknowledge A. Siehl, R. Alms, T. Jentzsch and A. Thomsen who introduced us into the 'secrets' of geological modelling. Special thanks go to N. Klein, W. Mueller and M. Hammel of the Computer Science Department, who made a significant contribution in the implementation of the GeoStore and GeoToolKit software. W. Skala (Berlin) and an anonymous referee read the manuscript.

References

Alms, R., Balovnev, O., Breunig, M., Cremers, A.B., Jentzsch, T. & Siehl, A., 1998. Space-Time Modelling of the Lower Rhine Basin Supported by an Object-Oriented Database. Physics and Chemistry of the Earth 23 (3). Elsevier Science Publishers (Amsterdam): 251-260.

Alms, R., Klesper, C. & Siehl, A., 1996. Three-Dimensional Modelling of Geological Features with Examples from the Cenozoic Lower Rhine Basin. *In:* Foster, A. & Merriam, D. (eds) Geological Modelling and Mapping. Plenum Press (New York): 113-133.

Balovnev O., Breunig, M.& Cremers, A.B., 1997. From GeoStore to GeoToolKit: The second step. Proceedings of the 5th International Symposium on Spatial Databases, LNCS, 1262. Springer Verlag (Heidelberg): 223-237.

Bode, T., Breunig, M. & Cremers, A.B., 1994. First Experiences with GeoStore, an Information System for Geologically Defined Geometries. *In:* Proceedings IGIS'94, LNCS 884. Springer Verlag (Heidelberg): 35-44.

Fellner D., 1996. Extensible Image Synthesis. *In:* Wisskirchen, P. (ed.): Object-Oriented and Mixed Programming Paradigms. Springer Verlag (Heidelberg): 7-21.

Jentzsch, T. & Siehl, A., 2002. Kinematic subsidence modelling of Lower Rhine Basin. Netherlands Journal of Geosciences / Geologie en Mijnbouw 81: 231-239 (this issue).

Polthier, K. & Rumpf, M., 1995. A concept for Time-Dependent Processes. *In:* Goebel et al. (eds): Visualization in Scientific Computing. Springer Verlag (Vienna): 137-153.

Rumbaugh, J. et al., 1991. Object-Oriented Modelling and Design. Prentice Hall (New Jersey): 500 pp.
Thomsen, A. & Siehl, A., 2002. Towards a balanced 3D kinematic model of a faulted domain – the Bergheim open pit mine-. Netherlands Journal of Geosciences / Geologie en Mijnbouw 81: 241-250 (this issue).