# Advancing DB4GeO

**M. Breunig, E. Butwilowski, D. Golovko, P. V. Kuper, M. Menninghaus and A. Thomsen**

**Abstract** The analysis of complex 3D data is a central task for many problems in the geo- and engineering sciences. Examples are the analysis of natural events such as mass movements and volcano eruptions as well as 3D city planning and the computation of 3D models from point cloud data generated by terrestrial laser scanning for 3D data analysis in various domains. The volume of these data is growing from year to year. However, there is no geo-database management system on the market yet that efficiently supports complex 3D mass data, although prototypical 3D geo-database management systems are ready to support such challenging 3D applications. In this contribution we describe how we reply to these requirements advancing DB4GeO, our 3D/4D geo-database architecture. The system architecture and support for geometric, topological and temporal data are presented in detail. Besides the new spatio-temporal object model, we introduce new ideas and implementations of DB4GeO such as the support of GML data and the new WebGL 3D interface. The latter enables the direct visualization of 3D

M. Breunig (✉) · E. Butwilowski · D. Golovko · P. V. Kuper · M. Menninghaus
Geodetic Institute, Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: martin.breunig@kit.edu

E. Butwilowski
e-mail: edgar.butwilowski@kit.edu

D. Golovko
e-mail: daria.golovko@kit.edu

P. V. Kuper
e-mail: kuper@kit.edu

M. Menninghaus
e-mail: mathias.menninghaus@kit.edu

A. Thomsen
Institute of Geosciences, Christian-Albrechts-Universität zu Kiel, Kiel, Germany
e-mail: athomsen@geophysik.uni-kiel.de

J. Pouliot et al. (eds), *Progress and New Trends in 3D Geoinformation Sciences*, Lecture Notes in Geoinformation and Cartography, DOI: 10.1007/978-3-642-29793-9_11,

database query results by a standard web browser without installing additional software. Examples for 3D database queries and their visualizations with the new WebGL interface are demonstrated. Finally, we give an outlook on our future work. Further extensions of DB4GeO and the support for the data management for collaborative subway track planning are discussed.

## 1 Introduction

The demand for modeling and handling large 3D and 4D data sets has been rapidly growing during the last decades (Breunig and Zlatanova 2011; Hashemi et al. 2009; Kolbe 2012; Kolbe et al. 2011; Mallet 2002; Raper 1989). Techniques and applications such as geodesy, terrestrial laser scanning (TLS), 3D city planning, geological modeling, geothermal reservoir modeling, and early warning of natural events strengthen this trend by generating large volumes of 3D data. However, the analysis of these data becomes a confusing task without the help of geo-databases, because the geo-expert has to access dozens or even hundreds of single data files. Furthermore, without documentation and long-time archiving of modeling and simulation (results) in a geo-database, many examinations become useless as soon as their authors are no longer available.

In this paper we argue that 3D geo-databases can be accessed by non-experts in a straightforward manner. The rest of this paper is organized as follows. In Sect. 2 the service-based system architecture of DB4GeO, our 3D geo-database kernel, is presented. Section 3 is dedicated to the geometric, topological, and temporal database support. Section 4 describes the support for GML data and implementation details of the REST communication interface. Section 5 presents the new WebGL interface of DB4GeO with various visualization examples. Finally, we give a conclusion and outlook on our future work concerning 3D and 4D geo-data support in DB4GeO, e.g. in the field of collaborative subway track planning.

## 2 System Architecture

DB4GeO (Bär 2007; Breunig et al. 2010; Thomsen et al. 2008b) has its roots in the development of GeoToolKit (Balovnev et al. 2004), an object-oriented library for 3D geometric data types for geo-databases. It has a service-based user interface and is exclusively implemented in the Java programming language. Hitherto REST (Fielding 2000) is used as communication platform, i.e., REST style web services are used to enable remote interaction of clients with the geo-database. The system architecture of DB4GeO is presented in Fig. 1. On the client side, GIS or mobile clients have access to 3D data managed by the DB4GeO server. On the server side, DB4GeO is accessed exclusively via its service infrastructure. The services are divided into simple and complex services (Breunig et al. 2010). The simple services

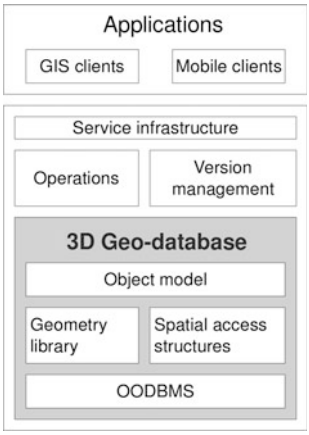


**Fig. 1** DB4GeO system architecture

are equivalent to basic geometric and topological operations such as the distance between 3D objects or the determination if two 3D objects intersect etc. However, also computations such as the intersection between two surfaces belong to the simple services. A typical complex service is the so called "3D-to-2D service" that has been introduced in Breunig et al. (2010). It computes a vertical profile section of a geological subsoil model by intersecting all existing surface objects with a vertical plane. Finally, it projects all wells within a specified distance onto the vertical plane. The geological model has to be bounded by a 3D bounding box. Further examples of complex services are the "4D-to-3D service", which calculates the geometry of a spatio-temporal object at a given moment in time, and a triangulation service that creates a triangle net from a set of points.

The core of DB4GeO is its 3D geo-database which is based on a geometry library and the R-tree based spatial access structures. DB4GeO is implemented upon the open source object-oriented database management system db4o (Paterson et al. 2006; Versant Corp 2012).

Developed from a system designed primarily for geological applications, DB4GeO concentrates in the first place on the modeling of spatial and temporal characteristics of objects and their parts. Support of semantics has received less attention in our research so far but is expected to gain importance in our future work. Although DB4GeO has not had many users lately, we hope that extending the data model and the functionality of the geo-database will extend its user community. DB4GeO primarily aims at geoscientists who need to store and process 3D and 4D objects represented with simplicial complexes or who want to combine such data with non-simplicial geometries.

## 3 3D/4D Database Support

As a database management system, the main task of DB4GeO is to store and manage large sets of geo-data in an efficient way for a long period of time without loss and free of contradictions (i.e., consistent data storage). DB4GeO provides a tightly defined, extensive set of geometric, topological and spatio-temporal objects that can be stored, managed and retrieved. Data models of these objects will be presented in the following subsections.

### *3.1 Geometry*

The object model of the geometric component in DB4GeO has been discussed in detail in Bär (2007). Roughly explained, the object model can be summarized as follows: at the root of the object model, a 3D object is located (`Object3D` class). Part of every 3D object is one three-dimensional spatial object (`Spatial3D`). The spatial objects can be of one of *four* abstract data types, namely either a *sample*, a *curve*, a *surface* or a *volume*. Any of the mentioned abstract data types has a concrete realization in the database. Currently, these are point nets (`PointNet3D`) as realization of the sample type, segment nets (`SegmentNet3D`) as realization of the type curve, triangle meshes (`TriangleNet3D`) as realization of the surface type and tetrahedral nets (`TetrahedronNet3D`) as realization of the volume type. All of these concrete classes for spatial data types are nets of the most simple geometric constructs (or geometric elements) of the respective dimension (so-called *simplices*).[1] A geometric net of any of these types in turn consists of an arbitrary number of disconnected net components. A net component is a contiguous geometric object, which consists entirely of geometric elements of one of the geometric types *point*, *segment*, *triangle* or *tetrahedron*.

### *3.2 Topology*

The topology model of DB4GeO extends its geometry model and is closely linked to it. The topology model provides an additional construct which allows to go beyond the model of simplicial complexes and to manage more complex structures. Furthermore, it enables easy and efficient navigation in the meshes. The topology model of DB4GeO is based on the combinatorial concepts of *Generalized Maps* (abbreviated as G-Maps) and *cell-tuple structure* introduced by Lienhardt (1989)

[1] The entire geometric model of DB4GeO is based on the model of simplicial complexes introduced in the context of GIS by Egenhofer and Moise, cf. Breunig (2001).

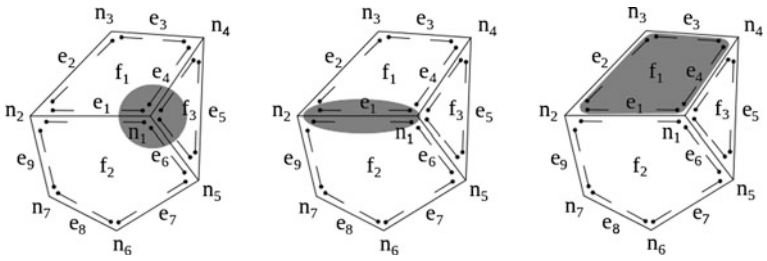


**Fig. 2** Fragment of a face net with cell-tuples shown as darts and orbits of a node (*left*), an edge (*center*) and a face (*right*)

and Brisson (1989), respectively. Further research on the theory of G-Maps can be found in Fradin et al. (2005); Lévy and Mallet (1999), and Thomsen et al. (2008a).

The G-Map representation itself does not provide information about the spatial extent of modeled objects. The link between the topology model and the geometry model of DB4GeO is realized by constructing the topology representation on top of an already existing `TriangleNet3D` or `TetrahedronNet3D`. The created object is represented by the classes `FaceNet3d` or `SolidNet3d`, respectively. Similarly to the geometry model, it includes one or more components. Instead of points, segments, triangles and tetrahedrons of the geometry model, the topology model manages four types of cells: nodes, edges, faces and solids. The cells are not limited to simplices. A cell-tuple represents a unique combination of a node, an edge, a face and a solid. Cell-tuples that differ in only one cell are linked to each other by so-called *involutions*. Specific combinations of involutions form so-called *orbits* used to define cells and groups of cells. Cell-tuples and orbits of various dimensions are shown in Fig. 2.

Figure 3 demonstrates the principal classes of DB4GeO topology model used to manage face nets. Since the concept of G-Maps is not bound to a particular dimension, the handling of solid nets was designed similarly to the handling of face nets without great difficulties. Each component of a `FaceNet3d` or a `SolidNet3d` consists of a net level ($C_{NL}$, class `FaceNet3dCompNetLevel`) and an object level ($C_{OL}$, class `FaceNet3dCompLevel`). Both classes implement the interface `CellNet3dCompLevel`. The topology of $C_{NL}$ exactly repeats the topology of the underlying triangle or tetrahedron net, i.e., every face or solid of $C_{NL}$ is a simplex and points to the corresponding simplex of the triangle or tetrahedron net. Each node of $C_{NL}$ is also linked to a particular point and can thus access its coordinates. $C_{OL}$ describes the overall geo-object structure that is modeled by cells that are commonly composed of a large amount of simplices (so-called "big cells"). At the creation of the topological net, $C_{OL}$ consists of exactly one cell whose boundary is identical to the boundary of the whole component. $C_{OL}$ is linked to $C_{NL}$ by the `higher` and `lower` attributes of their cell-tuples. The `higher` attribute of a cell-tuple of $C_{OL}$ points to a corresponding cell-tuple of $C_{NL}$, and it is `null` for every cell-tuple of $C_{NL}$. The `lower` attribute of a cell-tuple of $C_{NL}$ stores the link to its counterpart at $C_{OL}$ if such a cell-tuple

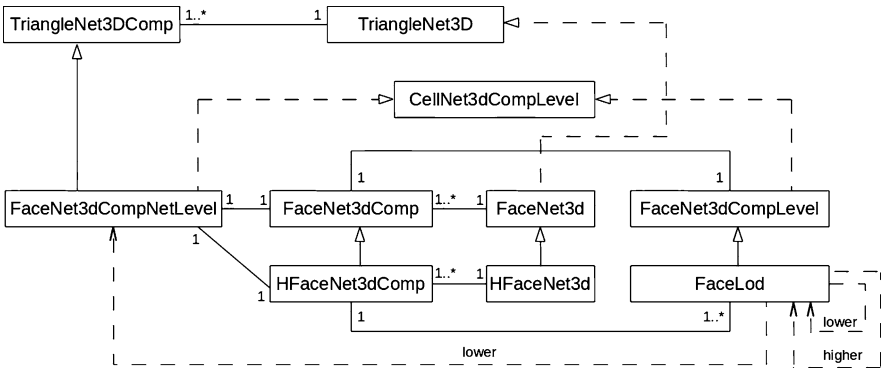


**Fig. 3** Classes of DB4GeO topology model used to manage face nets

exists; otherwise the `lower` attribute is `null`. A special `OrbitIterator` allows retrieving cell-tuples of $C_{NL}$ that are inside a particular cell of $C_{OL}$.

Such a structure permits separating geometry from topology, which is handy e.g. for the management of temporal changes that involve the object's coordinates but not its topology (cf. Sect. 3.3). Direct links to the geometry model enable access to the core functionality of DB4GeO at any time.

The topology module of DB4GeO offers the possibility to manage multiple levels of detail (LoDs) by using hierarchical G-Maps also referred to as HG-Maps (Lévy 2000; Fradin et al. 2005; Thomsen et al. 2008a). In DB4GeO, all LoDs have the same geometric extent and it is possible to relate particular locations of various LoDs to each other. This approach is different from e.g. the CityGML specification where various LoDs of the same object do not correspond to each other geometrically and can even be represented by different geometry types. For instance an object may be represented by a polygon at a more detailed LoD and by a line at a less detailed LoD. Furthermore, the geometry of objects in CityGML at each LoD is regarded as one entity and it is not always possible to determine which part of a more detailed LoD corresponds to a particular part of a less detailed LoD.

Similarly to cell-tuples, LoDs in DB4GeO also point to their `higher` and `lower` counterparts. The `higher` attribute of the most detailed LoD is $C_{NL}$. Note that, while a `FaceNet3dComp` has exactly one $C_{OL}$, an `HFaceNet3dComp` can have multiple LoDs (cf. Fig. 3). $C_{OL}$ can be regarded as one of the LoDs.

Two ways of creating a new hierarchical face net (`HFaceNet3d`) are available to the user. First, it can be created from an already existing non-hierarchical face net (`FaceNet3d`). In this case, the underlying triangle net and $C_{NL}$ of the original `FaceNet3d` are copied for the new `HFaceNet3d`, and $C_{OL}$ of the original net is converted to be an LoD of the new hierarchical geo-object. This permits retaining the current subdivision of $C_{OL}$. Secondly, the user can create a new hierarchical face net from a set of triangles, similarly to the construction of a non-hierarchical net. In that case, the underlying triangle net is created first, then the net level, and finally the first LoD of the new hierarchical net with exactly one face. In order to

ensure the geometric correspondence between LoDs, a new LoD can only be created as a copy of an already existing LoD. Besides, this allows the users to edit the hierarchical net more conveniently, because they can transfer already made changes to another LoD and continue editing.

The editing of LoDs in a `HFaceNet3d` and of $C_{OL}$ in a `FaceNet3d` is possible via 3D Euler operations. DB4GeO offers methods for inserting and removing nodes and edges of a face net. Before the changes are applied on the net, a check of constraints takes place. For both types of nets, topological correctness within the edited $C_{OL}$ or LoD has to be ensured. Hierarchical nets additionally require verifying that the hierarchy of LoDs is not violated, i.e., that every cell and every cell-tuple has its counterpart at the next more detailed LoD. If the constraints are fulfilled, the changes are carried out. Implementing 3D Euler operations and the management of hierarchies for solid nets are the objectives of future work.

## *3.3 Time*

The modeling of time in geoscientific information systems (and in solid modelers in general) is a subject of widespread research. One of the research issues is the combination of continuous temporal geometry models with discrete temporal topology models. Polthier and Rumpf added some relevant work on this issue. In Polthier and Rumpf (1995) they propose the concept of *adaptive time-dependent discretization*. Thomsen and Rolfs designed a concept to handle and store the vertices of time-dependent simplex nets efficiently. This approach was termed *Point Tube model* (Rolfs 2005, p. 51). Another important approach in this context is the concept of *Delta-Storage* used by Strathoff during the development of GeoToolKit to reduce the storage volume of redundant data between two timesteps (Strathoff 1999).

In DB4GeO we implemented a combination of these three concepts to support spatio-temporal data. Therefore it is possible to create time series of 3D objects whose topology changes with time. Due to the concept of *Delta-Storage* and the *Point Tube model*, we were able to reduce the amount of required memory and increased the performance of operations such as the interpolation between time steps or the generation of snapshots at specified dates (Kuper 2010). Figure 4 shows a 4D object containing 5 timesteps with a change of the net topology at timestep 3.

However, a restriction of the implemented model is that the topology of the net configuration of the geo-object has to stay invariant throughout all time steps–only the geometry may change. Also Polthier and Rumpf make no statement on how to model the transition between two states of the object's topology (object level) of the same geo-object. In such a transition, the geometry generally stays unchanged, but the topology of the geo-object as a whole may change. The implementation of the space-time module of DB4GeO also excludes the issue of managing change in spatio-temporal topology. Raza and Kainz have done some relevant research in the

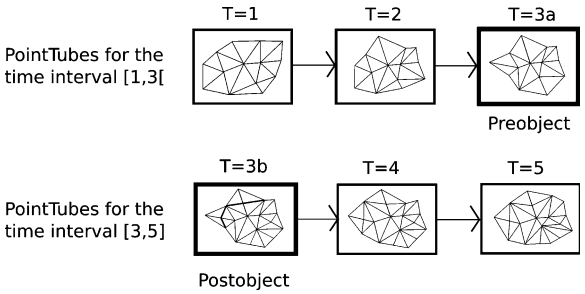


**Fig. 4** Representation of a 4D object with the new 4D model of DB4GeO (Kuper 2010)

area of spatio-temporal topology. In Raza and Kainz (1999) they propose a temporal cell-tuple structure to manage *Spatio-Temporal-Attribute Objects* in generic temporal GIS (TGIS). The *temporal cell-tuple structure* used by Raza and Kainz is also based on the cell-tuple structure of Brisson and thus is comparable to the internal model of the topology module for DB4GeO. In contrast to the concept of Raza and Kainz, we use a graph representation and not a relational approach to manage and store the cell-tuple data. We also combine the concept of temporal cell-tuple structure with our previously discussed concepts of spatio-temporal geometry and hierarchical G-Maps to manage the temporal topology of "big cells" (cf. Sect. 3.2). Figure 5 shows a simple example application sketching the intended functionality of the proposed spatio-temporal topology module for DB4GeO. This module is currently in implementation process.

Figure 5 shows a similar case as demonstrated in Fig. 4. Both illustrations depict a geo-object that moves through time and changes its geometry (in time intervals) and the topology of its meshing (at time steps). The example of Fig. 5 consists of three time steps. The geo-object is created at time step $t = 1$. Between the time steps $t = 1$ and $t = 2$ the geometry of the spatial object changes (the object grows). At time step $t = 2$ the meshing of the object changes, while the geometry remains constant. The geometry then changes again in the period between time steps $t = 2$ and $t = 3$. In the spatio-temporal model of DB4GeO this temporal geo-object is internally managed as two sequential `PointTubes` (that describe the point geometry at each interval) and two sequential temporal triangle nets (`TriangleNet4D`) (that describe the meshing at each interval). This means that whenever a change in the meshing takes place (here at $t = 2$), there is a break in the continuity of the spatial object. At this point, all object identifications get lost. From this point on, the evolution of the spatial object cannot be clearly traced. An unambiguous assignment of all geometric elements between the pre- and post-object of $t = 2$ would not be possible due to the changed meshing.

However, even if an unambiguous assignment of all geometric elements is not possible, still some elements can be assigned (cf. coincidences of pre- and post-objects of $t = 2$ in Fig. 5). Such conditions can be used to trace the mentioned temporal big cells. In Fig. 5, the thick lines symbolize the boundaries of two face cells (big cells) that are part of a face net component at object level $C_{OL}$. While `TriangleNet4D` $a$ ends at $t = 2$, the topology of the geo-object stays constant

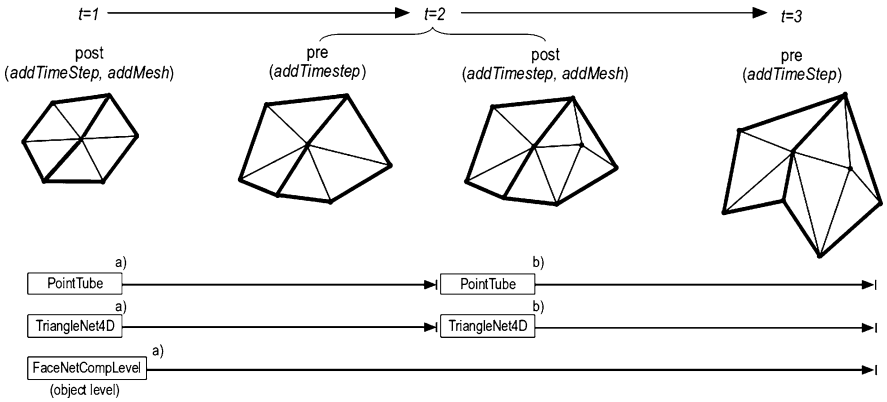


**Fig. 5** Simple example of a temporal cellular network of net level (*thin lines*) and of object level (*thick lines*)

at the object level (cf. lifespan of `FaceNetCompLevel` in Fig. 5), thus the two faces can be traced throughout the whole lifespan of the geo-object.

## 4 Web-Based Geo-Data Access

One of the objectives of DB4GeO is to remotely provide geo-database services. Nowadays, web-based access to geo-data is closely associated with OGC standards. In the following, we discuss the handling of GML data in DB4GeO and afterwards present the implementation of the REST-based architecture of the geo-database (Fielding 2000).

### *4.1 Support for GML Data*

Originally, DB4GeO was only capable of exchanging data in its own XML-format. However, as the OGC standards are developed and spread out among the providers and users of spatial information, the need for extending DB4GeO to offer data via OGC services is becoming more acute. To enable that, support for the Geography Markup Language (GML) is developed. This will make DB4GeO a handy tool that can at first import the results of 3D geomodeling software that does not offer OGC support, and then provide those results in a data format compatible with OGC standards. An example of such software is Gocad® (Gocad Research Group 2012), a widely used tool for 3D subsurface modeling. The geometry models of both DB4GeO and Gocad® are based on simplicial complexes making it easy to

exchange data between the two. DB4GeO offers importers and exporters for various types of Gocad® geometrical objects (GObj).

A further reason to offer GML support in DB4GeO is the growing number of interdisciplinary projects with joint handling of data from different sources, e.g. of nature-formed and anthropogenic objects. While DB4GeO originated in the field of geosciences, particularly that of geology, integrating CityGML data will make the database interesting for instance for subsurface construction projects where building and infrastructure data are processed and visualized together with the information about geological structures (Breunig et al. 2011).

In GML, objects are modeled as *features* that have geometry as a property. The geometry model of GML is based on boundary representations, e.g. polygons are defined via their `exterior` and `interior` (the latter is used when polygons have holes). The `exterior` and `interior` are represented by ordered lists of point coordinates. In our work, GML 3.0 geometries only with linear interpolation between points have been considered. Those lists can include an arbitrary number of points. Planarity assumed by many GML geometries is well suited for modeling anthropogenic objects that are geometrically simple. This simplicity also allows to construct more complex hierarchies of geometries. However, nature-formed objects in most cases are more compex and far from planar, so they have often been modeled in geosciences using simplicial complexes native to DB4GeO (Breunig 2001; Balovnev et al. 2004). Because of the different assumptions of the two data models representing anthropogenic and nature-formed objects, data in most cases cannot be transferred between them without adjustments of geometry.

On the one hand, integration of non-simplex geometries of GML into DB4GeO requires a representation by simplices. A method that triangulates complex planar polygons has been implemented in DB4GeO for that purpose. On the other hand, importing GML data into DB4GeO just by triangulating non-simplicial geometric objects causes data loss and distortion. The initial structure of the data might be lost and thus the attributes related to it. For instance, if the original object with $n$ polygons, each with the ?population density? attribute, is imported into DB4GeO, its polygons will be substituted with an even greater number of triangles, and the density attribute will lose its sense in the new geometric boundaries. Therefore, an additional construct is necessary to store the original geometric structure.

The topology model of DB4GeO provides such a construct (cf. Sect. 3.2). The G-Maps structure (Lienhardt 1989; Mallet 2002) enables managing non-simplex polygons (Thomsen et al. 2008a, 2008b). The geometric extent of the object is stored at the net level ($C_{NL}$) of the face net, enabling access to diverse geometric operations available in DB4GeO. The object level ($C_{OL}$) models how triangles of $C_{NL}$ are aggregated to the original non-simplex geometries of GML.

Figures 6 and 7 provide examples of importing CityGML data into DB4GeO. After the data set of a part of Berlin's downtown (Fig. 6) obtained in CityGML format was imported into DB4GeO, it is possible to export it into one of the formats supported by the geo-database. For example, the data set can be exported into the

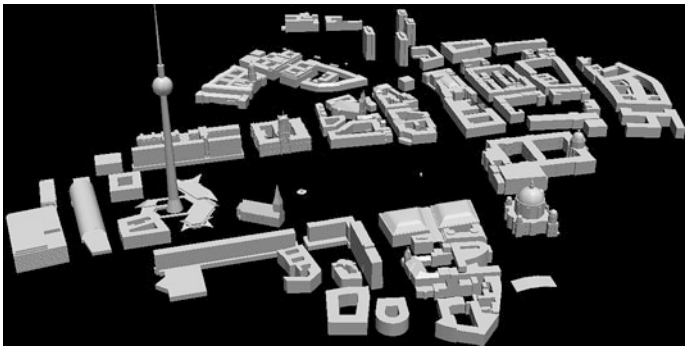

**Fig. 6** GML data (© City of Berlin, obtained from citygml.org) converted to Gocad® format in DB4GeO, visualized with ParaViewGeo®

Gocad® format (.ts) and visualized in Gocad® or another 3D visualization tool, e.g. ParaViewGeo® (ParaViewGeo 2012), together with data illustrating the subsurface of the area. Figure 7 looks closer at how a building can be stored in DB4GeO. The building came as a part of a CityGML data set. Its surface is represented by the GML geometry type `CompositeSurface`, which is made up of 17 `Polygons`. In DB4GeO, the non-simplex polygons were triangulated, which resulted in 32 triangles for the whole building. Those triangles become faces of $C_{NL}$. The 17 original polygons are represented in DB4GeO by faces of $C_{OL}$. This enables, for instance, assigning a certain color or a texture to each wall of the building (cf. Fig. 7).

While additional classes for the modeling of topology are available in GML 3.0, GML offers an alternative which is simpler and almost as powerful. GML uses the XML concept of `Xlinks` that reference resources via their IDs avoiding redundant data storage. For instance, if two neighboring buildings share a wall, the wall can be stored just once and then be references via an `XLink` from all other objects that use it. Such references carry information about the topology of the objects: first, the neighbor relationship between the two buildings is stored; secondly, the relationship between the wall and each of the buildings is defined (Krimmelbein 2011, p. 14ff). We have chosen to use the G-Maps topology concept of DB4GeO to represent non-simplex geometry types of GML rather than the less widely used GML 3.0 topology classes. In the future, we also plan to use `XLinks` to avoid redundancy when providing data from DB4GeO in the GML format. The implementation of G-Maps in DB4GeO already takes care of storing each node, edge, face and solid in the net just once under a unique ID, even if the original data structure stored them redundantly. The advantage of topology management is that this approach is dimension-independent, i.e., it can be applied to 0D-, 1D-, 2D- and 3D-geometries. Furthermore, topology is modeled explicitly without the semantic information of the objects.

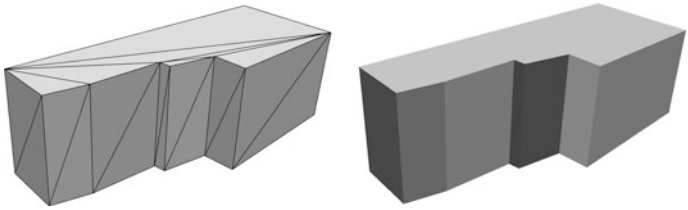

**Fig. 7** Building (© Ordnance Survey Great Britain, obtained from citygml.org) represented by faces of the net level (*left*) and of the object level (*right*), visualized with ParaViewGeo®

## *4.2 REST-Based Access and Query Examples*

DB4GeO is capable of providing its services via a RESTful web service architecture (Fielding and Taylor 2002). This architecture permits sending requests to the server via uniform resource identifiers (URIs). Each such request contains all the information necessary for the server to understand it and is independent of other requests that might have been sent to the server before. Such manner of communicating with the server is easier to interpret by humans without technical background, first, due to the familiarity of most people with URIs, and secondly, because of the isolation of requests. Furthermore, common browsers can serve as clients communicating with DB4GeO without installing additional software.

Manipulating resources using REST is possible via four commands: GET (retrieving a resource), POST (adding a new resource), PUT (updating an existing resource), and DELETE (removing a resource). DB4GeO uses two of them: GET and PUT. The latter can be used e.g. to add a new surface created by triangulation to the geo-database.

By default, when a request is sent to the geo-database, the response comes back in the DB4GeO-XML format and can be viewed in a browser. In DB4GeO, it is also possible to export the objects into other formats, e.g. the Gocad®, GML and VRML formats. In order to obtain an object in a certain format, the user should add the extension *.vrml*, *.gml*, *.ts*, *.vs*, *.so*, etc. at the end of the URI used to retrieve the object.

The REST-based services of DB4GeO are accessible from a web browser, via a Java application or an OpenJump plugin. In our future work, we plan to extend the number of operations available to the database users via the RESTful service and to create a user-friendly web interface to replace the XML-based representation in the browser.

Below are some examples of URIs used to query data from DB4GeO:

(1) http://server/projects/GeolProj
(2) http://server/projects/GeolProj/StructGeolSpace3D
(3) http://server/projects/GeolProj/StructGeolSpace3D/TestSurfaces
(4) http://server/projects/GeolProj/StructGeolSpace3D/TestSurfaces.ts

(5) http://server/projects/GeolProj/StructGeolSpace3D/TestSurfaces ?**intersects**(x1,y1,z1,x2,y2,z2)
(6) http://server/projects/GeolProj/StructGeolSpace3D/TestSurfaces ?**3dto2d**(CuttingPlane)
(7) http://server/projects/GeolProj/StructGeolSpace4D/TestSurfaces?**time**=0

Sets of objects in DB4GeO are managed by grouping them into spaces and projects. Spaces aggregate objects with the same dimension (e.g. 3D or 4D spaces), same coordinate reference, constraints, thematic information, etc. A project may contain multiple spaces. There is a defined way this hierarchical structure can be navigated via URIs. For example, the URI (1) returns information about the project *GeolProj*. That information includes the names of spaces that belong to the project. The user can easily add one of those names to the URI to access information about the corresponding space, like in the URI (2). In the same way, the user can go over to the object information in the XML format [(URL (3)]. If the object is to be retrieved in the Gocad® format for further processing in Gocad®, this is done by adding the extension *.ts* after the object name [(URI (4)].

Furthermore, it is possible to define operations to be carried out on objects via a URI. Operations include querying an object of a lower dimension, such as obtaining a 3D object for a certain point in time [(URI (7)] or the cross-section of a 3D object with a plane using the 3D-to-2D service of DB4GeO [(URI (6)]. Other operations include intersecting a given object with another object or a minimum bounding box [(URI (5)], projecting objects onto a plain and triangulating point sets.

## 5 Visualization with WebGL

Usually the visualization of 3D geodata takes place in various 3D modeling or visualization tools such as *Gocad*® or *ParaViewGeo*®. The user is responsible for the selection of one of these clients. For an alternative we developed a direct visualization of 3D database query results in a web browser based on WebGL (Khronos Group 2012).

WebGL (Web Graphics Library) is the mapping of OpenGL ES (Open Graphics Library for Embedded Systems) for web browsers and is now supported by almost all current browsers without installing an extra plugin. Since the graphical calculations runs directly on the hardware of the client, a high performance compared to traditional solutions such as VRML (Virtual Reality Modeling Language) and X3D (Extensible 3D) is provided. Thereby the user does not need to install any additional software apart from a modern web browser.

Thus with the help of our WebGL viewer it is possible to visualize geo-objects of the geo-database directly in a browser in 3D including lighting, colors, and controls.

For the visualization in WebGL we are using the free library *Three.js* (Three.js 2012). This JavaScript based library provides various types of cameras, lights and various shading concepts (e.g. Flat, Gouraud, Phong). We developed an exporter

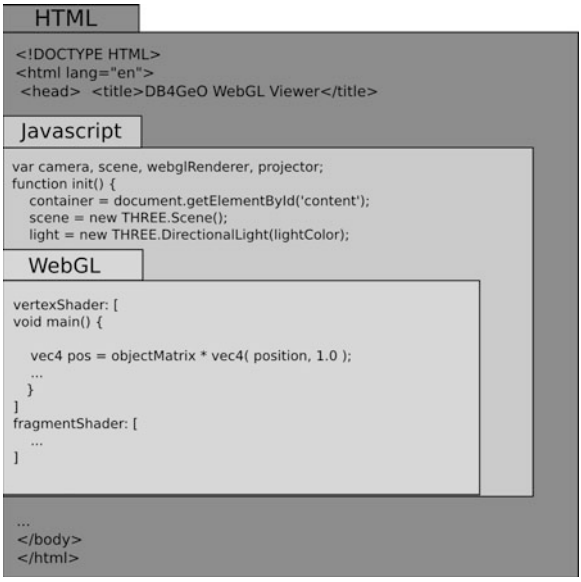


**Fig. 8** Structure of the HTML file generated by DB4GeO calling WebGL

which creates a 3D scene by using the geometry of the DB4GeO database. The following criteria were relevant for this intent:

- A suitable lighting.
- An intuitively controllable camera to view around the 3D object.
- Suitable shaders, colors, etc.

We decided to use `THREE.DirectionalLight` lights and a `THREE.PerspectiveCamera` camera. The DB4GeO WebGL Exporter creates an HTML file that executes JavaScript code with the use of *Three.js*. Within the JavaScript code different *Three.js* objects are created, adapted and transferred from the geometries of the spatial database into a format suitable for *Three.js*. For the DB4GeO objects `Point3D` and `Triangle3D` we use `THREE.Vertex` and `THREE.Face3`, respectively.

An overview of such an HTML file is shown in Fig. 8 and the result of an export in the Chrome® browser is shown in Fig. 9.[2] The viewer supports zooming, panning, and the possibility to rotate the object. The implemented WebGL viewer also shows the numbers of triangles for the visualized object. For instance the object demonstrated in Fig. 9 has 98,740 triangles.

The whole 3D model is transmitted from the database at the start of the viewer. Even larger amounts of data (test data sets with up to 2.5 million points) are represented efficiently due to the use of WebGL. In order to avoid further loading while viewing the model, caching is dispensed. To reduce the amount of data to be transferred, we plan to optionally provide a reduced 3D model.

[2] 3D-Model of "The Thinker" by Simon Schuffert (KIT), all rights reserved.

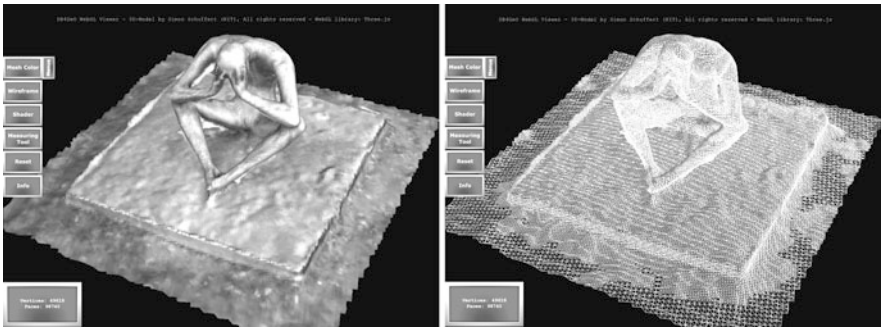

**Fig. 9** WebGL viewer of DB4GeO with colored mesh (*left*) and wireframe representation (*right*)

As one of the additional spatial operations the distance function has been implemented. In the future, various operations and queries should be directed back to the geo-database. After their executions, the results should again be displayed visually in the browser.

## 6 Conclusion and Outlook

In this contribution we have presented our last steps advancing DB4GeO, our service-based geo-database architecture. Besides the new object model for spatio-temporal data, new features are the support for GML data (implementation started) and the WebGL-interface (implementation completed) enabling direct geo-database access and visualizing 3D objects via a standard web browser without installing any additional software.

In our future work we intend to query geo-database operations directly from the WebGL viewer. Our focus here is on a BBox query, the query of meta-data, and the comparison of different time steps of a 4D object. Therefore we need to develop some additional spatio-temporal operations in DB4GeO and extend our WebGL viewer. Additionally we intend to examine how DB4GeO can be accessed via extended OGC services.

Finally, we intend to develop a new branch of DB4GeO supporting spatio-temporal data used for multi-scale subway track planning.

In the research group "Computer-Aided Collaborative Subway Track Planning in Multi-Scale 3D City and Building Models" (Breunig et al. 2011) a spatio-temporal database will be used to store several states and versions of building plans. As DB4GeO by now has been used to store geological data in 3D and time, it should be enhanced to support a data model that is commonly used to describe building models (Eastman 1999). Such parametric models stand in contrast to the simplicial complexes which are a core concept of DB4GeO. Re-using the simplicial model would lead to a loss of data when converting the parametric data into

the discrete triangle nets which can be stored by DB4GeO. Converting and storing the data in higher resolution would only increase the memory consumption, but not decrease the data loss in an equal manner. We will test two solutions. First, we intend to implement a hybrid model, which means that the database will convert the parametric geometries into discrete triangle nets handled by DB4GeO and we store the original file-based data linked to these converted geometries. By doing this we will be able to use the high performance queries of DB4GeO and its topology module. This does not cause any data losses, because both the parametric data is stored in the database and changes are simultaneously made on the converted geometries. Secondly, we intend to implement a complete new data-structure keeping the core concepts in mind. Therefore, we do not need to build the database from scratch and can use an already stable and proven system.

Another issue in advancing DB4GeO within the research group is the usage of spatio-temporal data within the database. In the geosciences, spatio-temporal data often consists of moving objects. Construction plans, however, may be modeled in two different ways. First, the temporal axis can show the construction progress of a building, i.e., it can be modeled which part of a building will appear at first and which one will appear later on. Therefore, the geometry of an object will not change in time, but only appear at a certain time step and may disappear later. Secondly, if several people develop a plan, there will be different versions of that plan, but it is unlikely to have one definite plan all the time. These different versions should be stored in the database in order to compare them with each other and re-use them later on, if needed. By using two qualities of time, *valid-time* and *transaction time*, we need to extend DB4GeO with a Bi-Temporal model (Worboys 1994). Experiences gained in handling the two qualities of time might be useful to integrate further time dimensions in DB4GeO later in the future. In order to perform efficient spatio-temporal queries, we also need to implement an efficient indexing technique. At first, we will extend the R*-Tree (Beckmann et al. 1990), which also is used in DB4GeO, to a Spatio-Temporal R-Tree (Saltenis and Jensen 1999). After that, we will concentrate on suitable and efficient query techniques (Snodgrass 1995).

**Acknowledgments** We thank Thomas Kolbe from the Technical University of Berlin, the City of Berlin and Ordnance Survey of Great Britain for the CityGML data sets. Furthermore, we are grateful to Simon Schuffert from Karlsruhe Institute of Technology for providing the 3D model of the figure “The Thinker”. This research has been funded by the German Research Foundation (DFG), grant no. BR2128/12-1 and BR2128/14-1.

## References

Balovnev O, Bode T, Breunig M, Cremers AB, Möller W, Pogodaev G, Shumilov S, Siebeck J, Siehl A, Thomsen A (2004) The story of the GeoToolKit–an object-oriented geodatabase kernel system. GeoInformatica 8(1):5–47 (Kluwer Academic Publishers, Hingham)

Bär W (2007) Management of geoscientific 3D data in mobile database management systems. In German. PhD thesis, University of Osnabrück, Germany

Beckmann N, Kriegel HP, Schneider R, Seeger B (1990) The r*-tree: an efficient and robust access method for points an rectangles. In: Proceedings of the ACM SIGMOD international conference on management of data (SIGMOD'90), pp 322–331

Breunig M (2001) On the way to component-based 3D/4D geoinformation systems. Springer, New York

Breunig M, Zlatanova S (2011) 3D geo-database research: retrospective and future directions. Comput Geosci 37(7):791–803. doi:10.1016/j.cageo.2010.04.016

Breunig M, Schilberg B, Thomsen A, Kuper PV, Jahn M, Butwilowski E (2010) DB4GeO, a 3D/4D geodatabase and its application for the analysis of landslides. Lecture notes in geoinformation and cartography for risk and crisis management, pp 83–102

Breunig M, Rank E, Schilcher M, Borrmann A, Hinz S, Mundani RP, Ji Y, Menninghaus M, Donaubauer A, Steuer H, Vögtle T (2011) Towards computer-aided collaborative subway track planning in multi-scale 3D city and building models. In: Proceedings of the 6th 3D geoinfo conference, p 17.

Brisson E (1989) Representing geometric structures in d dimensions: topology and order. In: Proceedings of the 5th ACM symposium on computational geometry, ACM Press, Washington, pp 218–227

Eastman CM (1999) Building product models, 1st edn. CRCPress, Taylor& Francis group, Boca Raton, Florida

Fielding RT (2000) Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine

Fielding RT, Taylor RN (2002) Principled design of the modern web architecture. ACM Trans Int Technol 2(2):115–150

Fradin D, Meneveaux D, Lienhardt P (2005) Hierarchy of generalized maps for modeling and rendering complex indoor scenes. Signal Image Communication laboratory, CNRS, University of Poitiers (Tech. rep.)

Gocad Research Group (2012) http://www.gocad.org. Accessed 29 Feb 2012

Hashemi L, Mostafavi MA, Pouliot J, Therrien R (2009) Developing an adaptive topological tessellation for 3D modeling in geosciences. J Can Inst Geom Geomat (GEOIDE students special issue) 63(4):419–431

Khronos Group (2012) http://www.khronos.org/webgl/. Accessed 17 Jan 2012

Kolbe TH (2012) CityGML. http://www.citygml.org. Accessed 12 Jan 2012

Kolbe TH, Konig G, König G, Nagel C (2011) Advances in 3D geo-information sciences. In: Lecture notes in geoinformation and cartography, Springer

Krimmelbein A (2011) Topologie in CityGML (Topology in CityGML). Diploma thesis, Karlsruhe Institute of Technology, Germany

Kuper PV (2010) Development of 4D object management for the geo-database DB4GeO. Diploma thesis, University of Osnabrück, Germany (in German)

Lévy B (2000) Computational topology: combinatorics and embedding. PhD thesis, National Polytechnic Institute of Lorraine (in French)

Lévy B, Mallet JL (1999) Cellular modeling in arbitrary dimension using generalized maps. http://alice.loria.fr/publications/papers/1999/gmaps/gmaps.pdf. Accessed 14 Dec 2011

Lienhardt P (1989) Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In: Proceedings of the fifth annual symposium on computational geometry, ACM Press, Washington, pp 228–236

Mallet J (2002) Geomodeling. Oxford Press, New York

ParaViewGeo (2012) http://sites.google.com/a/objectivity.ca/paraviewgeo/. Accessed 29 Feb 2012

Paterson J, Edlich S, Hörning H, Hörning R (2006) The definitive guide to db4o. Apress Series, APress

Polthier K, Rumpf M (1995) A concept for time-dependent processes. Visualization in Scientific, Computing, pp 137–153

Raper J (1989) Three dimensional applications in geographical information system. Taylor& Francis, London

Raza A, Kainz W (1999) Cell tuple based spatio-temporal data model: an object oriented approach. ACM-GIS, pp 20–25

Rolfs C (2005) Design and implementation of a data model for the management of 3D models in geoscientific applications. Diploma thesis, University of Osnabrück, Germany (In German)

Saltenis S, Jensen CS (1999) R-tree based indexing of general spatio-temporal data. Tech. rep, TimeCenter

Snodgrass RT (1995) The TSQL2 temporal query language, 1st edn. Kluwer Academic Publishers, Dordrecht

Strathoff F (1999) Memory-efficient management of time-dependent geometries for GeoToolKit. Diploma thesis, University of Bonn, Germany (In German)

Thomsen A, Breunig M, Butwilowski E (2008a) Towards a G-Map based tool for the modeling and management of topology in multiple representation databases. Photogrammetrie, Fernerkundung. Geoinformation (J Photogram Rem Sens Geoinf Process) 3:175–186

Thomsen A, Breunig M, Butwilowski E, Broscheit B (2008b) Modelling and managing topology in 3D geoinformation systems. In: Advances in 3D Geoinformation Systems. Springer, Heidelberg, pp 229–246

Three.js (2012) http://github.com/mrdoob/three.js/. Accessed 17 Jan 2012

Versant Corp (2012) Db4o. http://www.db4o.com. Accessed 12 Jan 2012

Worboys MF (1994) A unified model for spatial and temporal information. Comput J 37(1):26–34