Topological and Geometric Data Handling for Time-Dependent Geo-Objects Realized in DB4GeO

Martin Breunig, Edgar Butwilowski, Paul Vincent Kuper, Daria Golovko and Andreas Thomsen

Abstract In advanced spatio-temporal scenarios, such as the simulation of complex geo-processes, the analysis of complex surface- and volume-based objects changing their locations and shapes in time is a central task. For example, the documentation of landfills, mass movements or volcanic activities requires 4D modeling based on dynamic geometric and topological database structures. In this contribution we present our concepts and implementation efforts for the effective handling of geospatial and time-dependent data realized in DB4GeO, a service-based geo-database architecture. The topological and geometric data models of DB4GeO are described in detail. A geoscientific application of an open-pit mine demonstrates the usefulness of the concepts introduced at the beginning of the paper. Finally, an outlook is given on future geo-database work dealing with extensions of DB4GeO and the handling of geo-objects in the context of cooperative 4D metro tracks planning

M. Breunig (✉) · E. Butwilowski · P. V. Kuper · D. Golovko
Geodetic Institute, Karlsruhe Institute of Technology, Englerstr.7,
76131 Karlsruhe, Germany
e-mail: martin.breunig@kit.edu

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

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

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

A. Thomsen
Institute of Geosciences, Christian-Albrechts-Universität zu Kiel, Otto-Hahn-Platz 1,
24118 Kiel, Germany
e-mail: athomsen@geophysik.uni-kiel.de

S. Timpf and P. Laube (eds.), *Advances in Spatial Data Handling*,
Advances in Geographic Information Science, DOI: 10.1007/978-3-642-32316-4_1,

1 Introduction

The spatial data handling community looks back on a history of more than 25 years (Goodchild 1990; Marble DF 1984). During this time the geospatial data handling of volumetric 3D objects and their operations have been examined under different aspects by (Balovnev et al. 2004; Breunig 2001; Breunig et al. 1994; Döner et al. 2010; Mäntylä 1988; Pigot 1992; Pouliot et al. 2007, 2008, 2010; Schaeben 2003) and by other authors.

Based on this tradition and on the experiences of GeoToolKit, an object-oriented geo-database kernel for the management of 3D geometries (Balovnev et al. 2004), we have developed a geo-database architecture called DB4GeO (Bär 2007; Breunig et al. 2010). Right from the beginning, DB4GeO has been designed to support advanced geo-scenarios that require a web-based 3D/4D data access. It is implemented completely in the Java programming language and is based on the free object-oriented database management system db4objects (Versant 2011). The services of DB4GeO can be divided into primitive and complex services (Breunig et al. 2010). The primitive services contain basic geometric and topological operations such as calculating the distance between two objects, their relative position to each other (distinct, meet, overlap etc.), and the computation of the intersection between two objects. An example for a complex service is the "3D–2D service" that computes a profile section—i.e. the intersection of a vertical plane with 3D geometries—within a geological block model (Breunig et al. 2010). At the moment, the service architecture is implemented using REST (Fielding and Taylor 2002).

The paper is organized as follows. Sections 2 and 3 are dedicated to the concepts and implementations for spatio-temporal data handling in DB4GeO focusing on topological and geometric database support. In Sect. 4, a geoscientific application is presented. Finally, Sect. 5 gives an outlook on our future work.

2 Spatio-Temporal Concepts of DB4GeO

2.1 Geometric and Topological Core Model

The DB4GeO core API implements the simplicial complex model for the spatial part of its 3D object model.[1] The core API defines a 3D object to be an object in 3D space that has a spatial part which can be a *sample*, a *curve*, a *surface* or a *volume*. These abstract geometry concepts are specified by specific geo-objects as follows[2]:

[1] For an elaborate UML class diagram of the DB4GeO kernel geo-object model cf. Bär (2007), p. 65.

[2] For a visual overview of the geometry model of DB4GeO, cf. Butwilowski and Breunig (2010).

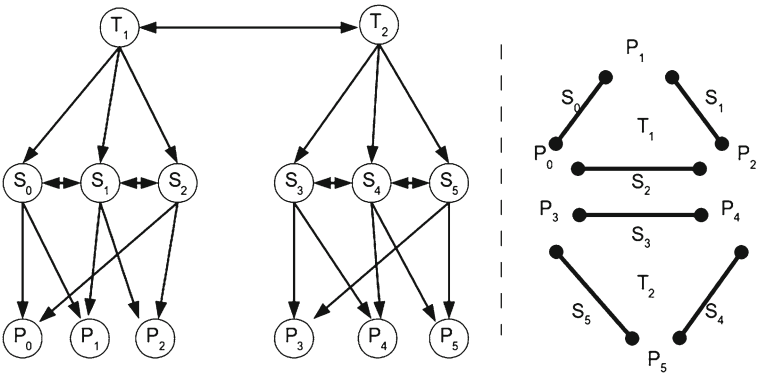

Fig. 1 Part of the incidence graph for simplicial complex model of the DB4GeO kernel

- sample as *point net*,
- curve as *segment net*,
- surface as *triangle net*, and
- volume as *tetrahedron net*.

The aforementioned nets are subdivided into disconnected *net components*. A net component itself consists of connected simplices (also *simple geo-object*). By means of the core API it is possible to navigate on top of net components by iterating over the explicitly stored *contact relations* between the simplices of the net.[3] Figure 1 shows a part of the incidence graph of the simplicial complex model as it is implemented in DB4GeO.

The arrows (left side) represent the connections between the simplices. Depicted is an example of two triangles T_1, T_2 that are adjacent by the segments S_2, S_3 (see right side). There are directed top-down incidence relations *from triangles to segments to points* as well as "next to" connections between multiple triangles and between multiple segments. This incidence graph is commonly used in geometry modeling systems (Lévy and Mallet, 1999, cf. p. 3), but obviously there are also some insufficiencies concerning the navigational properties of this structure. For example, there are no back references from lower to higher dimension simplices as well as there is e.g. no direct connection between S_2 and S_3, which makes navigation quite difficult. In some cases, it is necessary to traverse the whole structure to do one step in navigation.[4] While this model is suitable for many applications, it also has some shortcomings. For example, it is not possible to distinguish *subdivisions* on a net component, i.e. cells (e.g. faces/volumes) composed of multiple simple geo-objects (triangles/tetrahedrons) forming a net component. A potential improvement of this shortcoming will be discussed in Sect. 2.3.

[3] This structure can be seen as the implicit topology model of the DB4GeO/DB3D core API.

[4] For example, if it is necessary to find all neighboring segments to a given point.

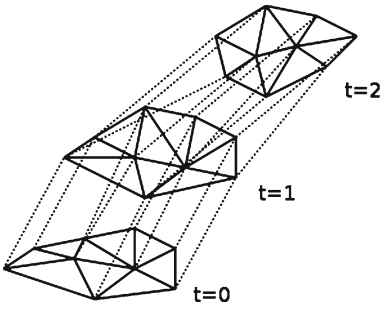

Fig. 2 Representation of a 4D object (Rolfs 2005)

2.2 A First Geometric 4D Model

To support spatio-temporal data handling in DB4GeO, a 4D model on top of the described geometric core model has been developed (Rolfs 2005). Due to this first 4D model, DB4GeO was able to handle simplex-based data within a fixed time interval. Such a 4D object has a spatial part which can be a sample, a curve, a surface or a volume (cf. 3D object) and is located in a 4D space. Figure 2 shows the representation of such a 4D object, which is comparable to the representation of Worboys's spatio-temporal model (Worboys 1992).

Due to the internal implementation, the first geometric 4D model of DB4GeO is not able to extend the time interval of an existing 4D object with further data. Furthermore, the data needs to meet several constraints to work properly. The major constraints are:

- The import format is proprietary;
- The net topology cannot change within the time interval;
- The sequence of time intervals is fixed and cannot be modified.

According to these constraints a new 4D model to handle spatio-temporal data was developed in Kuper (2010). The techniques to improve the functionality, user experience, and performance are described in Sects. 2.4 and 3.2.

2.3 Topological Structure

For the construction of regions by aggregation of multiple triangles or tetrahedra, following a naive approach, it would be sufficient to assign to every individual triangle or tetrahedron of the simplicial complex an attribute that determines to which region the respective simplex belongs. However, with this naive approach it would not be possible e.g. to navigate along the edge geometries of the regions (for example along the boundary surface between two volumes or along the boundary segment between two surfaces) efficiently. That is why DB4GeO handles topology in a more

generic way following works of the 3D modeling community by Lienhardt (1989), Brisson (1989), and Lévy and Mallet (1999).

Brisson and Lienhardt proposed explicit generic topology models that address *cell-tuple structures* and *Generalized Maps*, respectively. The way to the cell-tuple structures and the Generalized Maps was paved by prior topological models that have widely been used in the CAD community, such as the *winged edge representation* of Baumgart and the *half-edge structure* and *radial edge representation* of Weiler (Baumgart 1975; Weiler 1985, 1986). The cell-tuple structure has finally been proposed by Brisson (Brisson 1989).

Brisson introduced the notion of *cell-tuple* that is defined as an ordered sequence of cells of decreasing dimension: a cell-tuple corresponds to a path in the incidence graph (cf. Fig. 1). The cell-tuples are "connected" through the concept of adjacency that is inherent to the cell-tuple structure: two cell-tuples C and C' are called i-adjacent (A_i) if exactly one cell, namely the cell of dimension i of the cell-tuple is exchanged (so called *switch* operation) so that another tuple of the set of valid cell-tuples (a *permutation*) is obtained in return.

Lienhardt proposed the d-Generalized Map, (d-G-Map), a more abstract model of the topology of a d-dimensional cellular complex, based on algebraic topology (Lienhardt 1989). A *d-G-Map* is defined as a pair of a set of *darts* and of a set of $d+1$ *involutions*, noted α_i, i.e. transformations defined on the darts verifying $\alpha_i \alpha_i = id$ for $i = 0, 1, \ldots d$. Moreover, for any pair of indexes i, j with $j = i + 2 + k$, $\beta_{ij} = \alpha_i \alpha_j$ again is an involution, which implies that α_i and α_j commute. With this structure, a d-G-Map can be interpreted as a special *abstract simplicial complex*. A possible representation of a d-G-Map is a d-celltuple structure. Another possible representation is a graph *G(D,A)* with the set D of darts as nodes, and the set A of edges composed of d classes of pairs of darts defined by the $d + 1$ involutions $\alpha_0 \ldots \alpha_d$.

A particular class of G-Maps are *orientable d-G-Maps*. These can be represented by bipartite graphs with two classes of darts with opposite "polarity", linked by the edges defined by the α_i involutions. For our practical applications in geosciences, only orientable G-Maps are considered. A spatial model of a d-G-Map (or of a celltuple structure) is obtained by an *immersion* into the euclidean space R^d. Thus different spatial models can be derived from the same G-Map by applying different *immersions*.

2.4 Advanced Geometric 4D Model

The new model to handle geometric spatio-temporal data in DB4GeO improving the first model introduced in Sect. 2.2 is based on three main techniques with the following objectives:

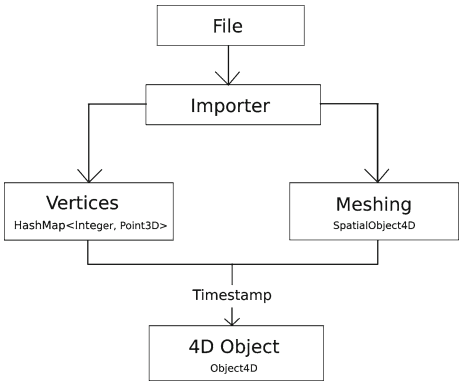

Fig. 3 Workflow adding a time step to a 4D object in DB4GeO (Kuper 2010)

- PointTubes: to handle and store the points of a time-dependent 2–3D simplex net efficiently;
- Pre- and post-objects (Polthier and Rumpf 1995): to offer a solution for changing net topology within time;
- Delta-storage (Strathoff 1999): to avoid redundant storage of points and topology information.

The spatio-temporal handling of data is focused on moving vertices in a 3D space within a time interval. DB4GeO handles the information about the net topology separately from the vertices. Due to the implementation of the spatio-temporal model presented in Polthier and Rumpf (1995), the net topology can change in time. At every time step, a pre-object and a post-object exist. The pre-object of time step t_i and the post-object of time step t_{i-1} have the same net topologies, i.e. discretizations. The pre-object and the post-object of one time step have the same geometry, i.e. the location of the vertices, while their net topologies can differ. The discretization can change at every time step t_i while the geometry can change between two time steps.

DB4GeO provides two main functions to build a proper 4D object: *addTimestep()* and *addTopology*. The former function adds the information about the vertices while the latter adds the net topology. The topological information about the net structure is added to the 4D object with every change of the topology, i.e. in a time step with a pre- and post-object. Therefore, DB4GeO is able to interpolate between time steps without paying attention to the net topology. It interpolates the vertices stored in PointTubes and uses the applicable topology when needed.

The time interval is extensible, i.e. it is possible to add additional time steps to an existing 4D object with a simultaneous update of the end date of the time interval. To improve the performance and to reduce the amount of storage data, DB4GeO only stores those objects of every new time step which contain new information, i.e. only the changes are stored. Every insertion of a new time step compares the vertices to the last one. Figure 3 provides an overview of the new 4D model workflow.

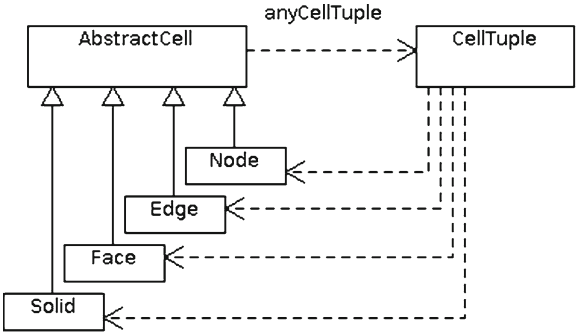

Fig. 4 References between abstract cell, cells and cell-tuples

3 Implementation of the Spatio-Temporal Concepts

3.1 Implementation of the Topology Concept

The G-Maps module for the management of the topology for 3D geo-objects in DB4GeO conceptionally relies on the notion of cells as a means to describe a geo-object by its decomposition. In the context of DB4GeO a cell may be any object that is composed of a simply connected set of simple geo-objects, e.g. a curved surface or a polyhedron or a solid. From a software modeling perspective, the conjunction between the classes of simple geo-objects, as they are implemented in the DB4GeO core API, and the cell classes, as provided by the G-Maps module (i.e. the conjunction between DB4GeO and the G-Maps module), are depicted and described in Butwilowski and Breunig (2010). Thus, the next step is to enable a connection of the cell classes to a cell-tuple class. This model is represented in Fig. 4, where the topological cell classes *Node*, *Edge*, *Face* and *Solid* (these are supported by the G-Maps module) are generalized by an `AbstractCell` class that has a reference `anyCellTuple` to a `CellTuple` class. Since all cell classes are of type `AbstractCell`, any cell "has a" cell-tuple. This cell-tuple represents an arbitrary cell-tuple of the cell. Conversely any cell-tuple has separate references to all four cells (cf. Fig. 4). This allows an unfettered back-and-forth navigation between the objects of all the cell types and the respective cell-tuple objects.

The class diagram considers the definitions of cell-tuple structure and G-Maps of Brisson and Lienhardt and serves as a basis for the innermost kernel of the G-Maps module for DB4GeO. An object of the `CellTuple` class is a composition of `Cell` objects of different dimensions[5] (which represent the incident cells). The field of each `CellTuple` object includes references to exactly four `CellTuple` objects

[5] i.e. of objects of the classes *Node*, *Edge*, *Face* and *Solid*.

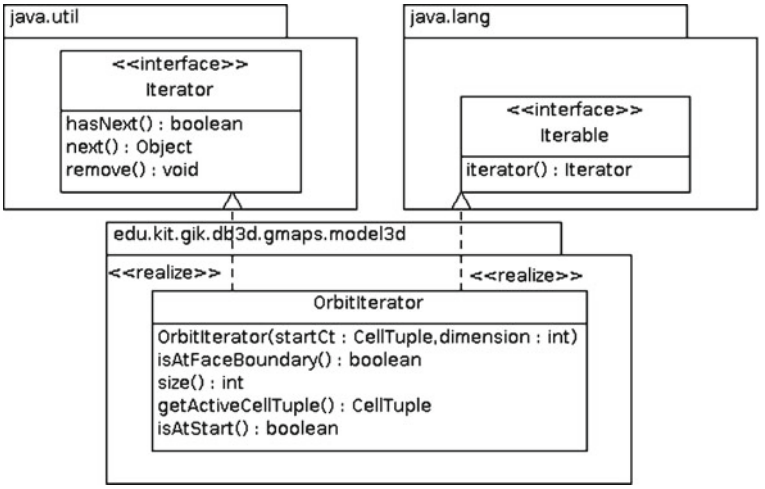

Fig. 5 Diagram of *OrbitIterator* class

called `alpha0`, `alpha1`, `alpha2`, and `alpha3`.[6] These references provide the means to perform fundamental/basic topological operations to navigate between the cells in any direction (these are the above mentioned G-Map operations).

These fundamental operations can be combined to more complex topological operations that are capable of traversing whole cells (so called *orbits*, cf. (Lévy and Mallet 1999, p. 5)). Orbits fit well with the concept of *iterators* (key concept of the Java programming language) which are defined by the interface `java.lang.Iterator`.[7] In our implementation, an orbit is represented by an `OrbitIterator` class that realizes the `Iterator` and the `Iterable` interfaces of Java (cf. Fig. 5).

Thus an `OrbitIterator` provides itself through its `iterator()` method and may directly be used in a `for`-loop. As a realisation of the `Iterator` interface, the `OrbitIterator` provides the methods `hasNext()` and `next()`.[8] For the instantiation of an `OrbitIterator` object, its constructor needs an object of type `CellTuple` (as its constructor parameters) which will be the start cell-tuple (*startCt*) of the orbit (this can be any cell-tuple of a cell in fact) and an integer value that defines the *dimension* of the orbit (cf. Fig. 5). As a result, a complete 0-dimensional orbit traversal around the node with dart `startCt` (i.e. the traversal of an orbit that would "collect" all cell-tuples around that node) is as simple and as elegant in our Java implementation as in the following example:

[6] These are not illustrated in Fig. 5 to reduce the diagram's complexety.

[7] Though, to be more precise, orbits are not only iterators but *circulators* (Devillers et al. 2011).

[8] The `OrbitIterator` does not provide a `remove()` method (in conformity with the orbit definition).

```
for(CellTuple ct : new OrbitIterator(startCt, 0))
{
  System.out.println(ct);
}
```

In our implementation, all connected cells of a cell net are summarized in a cell net component. Every cell net component is further subdivided into two *cell net levels*, i.e. a cell net component at *network level* or just *net level* (C_{NL}) and a cell net component at *object level* (C_{OL}). The topology of C_{NL} is an exact reproduction of the topology of the net structure of the underlying simple geo-objects net (i.e. of the simplicial complex). This level is mainly used for navigation purposes. It eases the algorithmic navigation on the net structure. C_{OL} on the other hand is the boundary representation of the component object (i.e. representing the whole geo-object itself). The topology defined by the cell-tuple structure cannot be edited by the user on C_{NL}, only on C_{OL}.

3.2 Implementation of the Geometry Concept for Time-Dependent Objects

For an efficient and user-friendly behavior of our 4D model we developed a simple API to work with 4D objects in DB4GeO. The main concepts mentioned in Sect. 2.4 are implemented in the class *object4D*. All three concepts work automatically. The implementation is based on two main functions:

addTimestep: the function is called with two parameters, the *vertices* which extend or create the PointTubes of an 4D object and a *java.util.Date* object to specify the time stamp. If there are any vertices already existing in the last time step, the internal PointTubes will be extended with references to these. Otherwise new *Point3D* objects for the extension of our PointTubes are created.
addTopology: the function is called with one parameter, representing the net topology of the just added time step. This object is called *SpatialObject4D* and consists of 0–n *Point4DNet*, *Segment4DNet*, *Triangle4DNet* or *Tetrahedron4DNet* objects. This information will be added to the 4D object. The number of *SpatialObject4D* objects will only increase if the last time step was a post-object, i.e. the *addTimestep* function was called twice with the same *Date* object.

To access single states of 4D objects at arbitrary dates we developed a class called *ServicesFor4DObjects*. This class contains one main function:

***getInstanceAt(Object4D, Date)*:** this functions creates a 3D object, i.e. a snapshot of the 4D object at the specified date via interpolation. The date must be part of the time interval. Due to the use of the implemented spatio-temporal model introduced in Polthier and Rumpf (1995), the computation of such a snapshot always refers to an interpolation between two sets of vertices in a *1:1* relation.

4 A Geoscientific Application Example

One of the use cases for DB4GeO has been the so-called Piesberg application. Piesberg is a hill near the city of Osnabrück in Northern Germany that has been exposed to open-pit mining for several decades. In the 1970s, its older part began to be used as a landfill causing changes to the volume and surface of the hill. A dataset representing the surface changes of Piesberg in 12 time steps between 1976 and 1993 is available (Lautenbach and Berlekamp 2002).

The present application example demonstrates how the topology module of DB4GeO can be used for a land use classification. Different classes of land use on Piesberg can be identified dividing the surface of the hill into five regions: "mining" (has a hole), "wind energy", "old landfill", "active landfill", "compost". Managing non-simplex regions necessary for this kind of applications would not have been possible using the geometry model of DB4GeO alone.

The G-Maps module of DB4GeO offers four operations for editing the composition of a face net: `insert node`, `remove node`, `insert edge`, and `remove edge`. The `insert edge` operation enables inserting both simple edges (with only two endpoints) and multi-edges (with more than two vertices). Before each of the operations is carried out, constraints are checked to avoid an invalid result. Examples of insertion and removal operations are shown in Fig. 6.

To simulate the Piesberg landfill, at first a face net with a net level and an object level is created. The net level represents the geometry of the object, in this case via a triangle net. The object level initially has one face; its boundary coincides with the boundary of the whole object. Then, we used the insertion and removal operations implemented in DB4GeO to divide the surface of the hill into the five land use classes. Notice that one of the five resulting regions of the object level has a hole. Finally, we used the `OrbitIterator` (cf. Sect. 3.1) to find faces of the net level that correspond to each face of the object level. Since the faces of the net level are simplices, we easily exported them from our geo-database into the gOcad® format (.ts) and visualized them in ParaViewGeo® (cf. Fig. 6).

Our next research goal is to extend the G-Maps module of DB4GeO to handle temporal changes and test it with all of the 12 available datasets of Piesberg.

5 Conclusion and Outlook

In this contribution we have presented concepts and implementations for geospatial and time-dependent data handling of complex geometric and topological objects realized in DB4GeO, our service-based geo-database architecture. Advanced topological and time-dependent geometric data models have been introduced. In our future work we will also focus on the straight-forward visualization of database results via WebGL from standard Internet browsers. Furthermore, the management

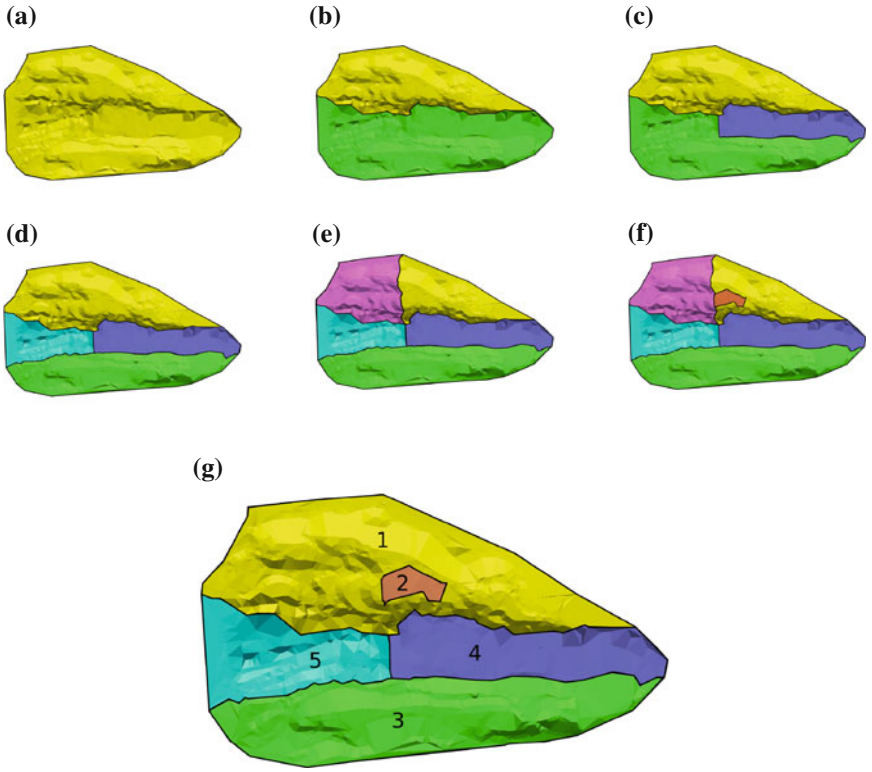

Fig. 6 Insertion and removal operations on Piesberg dataset (visualized with ParaviewGeo®) **a** Befor editing. **b** Edge inserted. **c** Edge inserted. **d** Edge inserted. **e** Edge inserted. **f** Two nodes and an edge inserted. **g** Two edges removed. Land use classes : 1- mining, 2- wind energy, 3- old landfill, 4- active landfill, 5- compost

of city models in DB4GeO will be examined using GML data. Another open question is how DB4GeO can be adapted to OGC Web services. Finally, we intend to examine the handling of 3D geo-objects used for the cooperative planning of metro tracks. Therefore, spatial representations other than simplicial complexes such as boundary representation or parameterized geometries should be directly supported by the geo-database.

Acknowledgments We thank Jürgen Berlekamp from USF, Osnabrück University and the Survey Office of Osnabrück city for the permission to use the Piesberg dataset for scientific purposes.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

Bär W (2007) Verwaltung geowissenschaftlicher 3D daten in mobilen datenbanksystemen. Ph.D thesis, University of Osnabrück, Germany

Baumgart BG (1975) A polyhedron representation for computer vision. In: Proceedings of 1975 national computer conference, AFIPS, pp 589–596, 19–22 May 1975

Breunig M (2001) On the way to component-based 3D/4D geoinformation systems. Lecture notes in earth sciences, vol 94. Springer, Heidelberg, p 199

Breunig M, Bode T, Cremers AB, (1994) Implementation of elementary geometric database operations for 3D-GIS. In: Waugh T, Healey R (eds) Proceedings of the 6th international symposium on spatial data handling, Edinburgh, pp 604–617

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 geoinformation and cartography risk crisis management, pp 83–102

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

Butwilowski E, Breunig M (2010) Requirements and implementation issues of a topology component in 3D geo-databases. Poster Abstract. In: Proceedings of the 13th AGILE international conference on geographic information science

Devillers O, Kettner L, Pion S, Seel M, Yvinec M (2011) Handles, ranges and circulators, September 2011. http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Circulator/Chapter_main.html. Accessed 06 Jan 2012

Döner F, Thompson R, Stoter J, Lemmen C, Ploeger H, van Oosterom P, Zlatanova S (2010) 4d cadastres: first analysis of legal, organizational and technical impact–with a case study on utility networks. Land Use Policy 27:1068–1081

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

Goodchild MF (1990) Keynote address: spatial information science. In: Proceedings, fourth international symposium on spatial data handling, Vol 1. Zurich, pp 13–14

Kuper PV (2010) Entwicklung einer 4D objekt-verwaltung für die geodatenbank DB4GeO. University of Osnabrück, Germany, Diploma thesis

Lautenbach S, Berlekamp J (2002) Data set for the visualization of Piesberg Landfill in Osnabrück. University of Osnabrück, Germany Institute of Environmental Systems Research

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

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

Mäntylä M (1988) Introduction to solid modeling. Computer Science Press, Rockville

Marble DF (ed) (1984) In: Proceedings, international symposium on spatial data handling, Zürich (1984) Geographisches Institut. Universität Zürich-Irchel, Abteilung Kartographie/EDV

Pigot S (1992) A topological modeling for a 3D spatial information system. In: Proceedings of the 5th international symposium on spatial data handling, Charleston, South Carolina, pp 344–360

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

Pouliot J, Badard T, Desgagné E, Bédard K, Thomas K (2007) Development of a web geological feature server (WGFS) for sharing and querying of 3D objects. In: van Oosterom et al. (eds) Advances in 3D geoinformation systems–Lecture notes in geoinformation and cartography, pp 115–130

Pouliot J, Bédard K, Kirkwood D, Lachance B (2008) Reasoning about geological space: coupling 3D geomodels and topological queries as an aid to spatial data selection. Comput Geosci 34(5):529–541

Pouliot J, Roy T, Fouquet-Asselin G, Desgroseilliers J (2010) 3D Cadastre in the province of quebec: a first experiment for the construction of a volumetric representation. In: Kolbe GNC, Knig TH (eds) Advances in 3D geo-information sciences. Lecture notes geoinformation and cartography

Rolfs C (2005) Konzeption und implementierung eines datenmodells zur verwaltung von zeitabhängigen 3D-modellen in geowissenschaftlichen anwendungen. University of Osnabrück, Germany, Diploma thesis

Schaeben H, Apel M, Boogaart KGvd, Kroner U, (2003) GIS 2D, 3D, 4D, nD. Von geographischen zu geowissenschaftlichen informationssystemen. Informatik-Spektrum 26(3):173–179

Strathoff F (1999) Speichereffiziente verwaltung zeitabhängiger geometrien für ein geotoolkit. University of Bonn, Germany, Diploma thesis

Versant (2011) http://www.db4o.com. Accessed 22 Nov 2011

Weiler K (1985) Edge-based data structures for solid modeling in curved-surface environments. Comput Graph Appl 5(1):21–40

Weiler K (1986) The radial edge structure: a topological representation for non-manifold geometric boundary modeling. In: Proceedings of the IFIG WG 5.2

Worboys MF (1992) A model for spatio-temporal information. In: Proceedings of the 5th international symposium on spatial data handling. Vol 1, pp 602–611