INTEGRATING QUALITY MANAGEMENT INTO A 3D GEOSPATIAL SERVER

Volker Coors[a] and Michel Krämer[b]

[a] Hochschule für Technik Stuttgart, University of Applied Sciences,
Schellingstr. 24, 70174 Stuttgart, Germany – volker.coors@hft-stuttgart.de
[b] Fraunhofer Institute for Computer Graphics Research (IGD),
Fraunhoferstr. 5, 64283 Darmstadt, Germany – michel.kraemer@igd.fraunhofer.de

Commission IV/8

KEY WORDS: GIS, Databases, Detection, Correction, Geometric, Algorithms, Urban, Requirements

ABSTRACT:

In recent years the technology and workflow for producing and management of large 3D urban models has been established and widely been used. Standards such as CityGML enable the modelling and exchange of semantically enriched multi-purpose 3D urban models for applications like urban planning, public participation, environmental simulation and navigation. However, data quality management is essential to control and enhance the quality of these models in order to be able to meet the needs of the aforementioned applications. Quality management should be performed throughout the whole lifecycle of geospatial datasets—from data acquisition to processing, analysis and visualisation. In this paper, we therefore focus on the integration of a quality management software module into a 3D geospatial data server. First results of a prototype system developed at HFT Stuttgart together with Fraunhofer IGD will be presented in this paper as a starting point for further research into the field of quality management of 3D city models.

1 INTRODUCTION

As 3D city models are more and more used for different applications like urban planning, energy management or disaster and emergency management, the need for high-quality data becomes apparent. The various users of city models have different understandings of the term *high-quality*[1]. For example, in a tourism application it is crucial that there is a high coverage of textured models, whereas this aspect is rather irrelevant for noise emission calculation and the like. Nevertheless, the different application areas also have common requirements: they need data that is up to date, complete and very accurate. In the past, geospatial data was kept in various data sources with very heterogeneous structures and formats. This made it hard to use them for more than one application. Apart from that, the different datasets were typically not stored in a common place. This led to a state where a lot of information is stored redundantly with diverging quality—in terms of up-to-dateness, completeness and accuracy.

3D geospatial data management systems mitigate this problem by managing city models and related information in a common database. They also provide means to keep the dataset up to date; mostly by providing a graphical user interface with the capabilities of adding objects as well as replacing existing models with higher detailed ones.

Most geospatial data management systems also support standardized OGC web services and data exchange formats. This enables a wide range of software products to access the data in the common database and to use it for specialized applications. In such cases, the geospatial data management system should make sure data fulfils the user's requirements. Hence, data quality management should be an integral part of the system, controlling quality throughout the whole lifecycle of the 3D city model: from data acquisition to actual usage.

[1] According to the ISO 9000 standard, *quality* is the "degree to which a set of inherent characteristics fulfils requirements" (TC 176/SC, 2005). Since requirements are defined by the user, the *quality* of a dataset highly depends on the user's needs and how it fulfils them.

Besides the quality aspects already mentioned—up-to-dateness, completeness and accuracy—data sets must also be free of geometrical problems, especially for applications where data is used for simulations and analyses based on mathematical algorithms. As a first step of combining quality assurance processes and geospatial data management, we present in this paper our results of integrating a quality assurance component into a 3D geospatial server. We focus on geometrical problems since we believe them to be of crucial importance for a wide range of applications. The quality management component has been developed by the HFT Stuttgart. We integrated them into the CityServer3D, a 3D geospatial data management system developed by the Fraunhofer IGD. In this paper we will first summarize the mathematical background. We will then present how the integration of the software components was performed. The paper concludes with the results of analysing a dataset from a real-world example.

2 RELATED WORK

The need for quality management for geospatial data has long been recognized. In 1995, Guptill and Morrison described the seven elements of spatial quality: Lineage, Positional Accuracy, Attribute Accuracy, Completeness, Logical Consistency, Semantic Consistency and Temporal Information (Guptill and Morrison, 1995). These elements are rather general—they do not depend on a special data model or dimension—and so they have been widely applied to other areas as well. For example, MacEachren et al. use the elements in the area of uncertainty visualisation. They also add qualitative attributes like *credibility* of the data supplier, *subjectivity*, and others (MacEachren et al., 2005). Van Oort uses the elements to perform fit-for-use assessment of datasets (Van Oort, 2005). He also extends the list by adding qualitative attributes like the *usage* and the *purpose* of the respective dataset. Apart from that, he states that the advantages of assessing and managing spatial data quality metadata has widely been recognized, but not yet brought to application—probably due to obstacles in implementations or workflows.

The works above deal with spatial data in general, but they do not focus on the specifics of 3D city models. The additional third dimension makes the problem of quality assessment in the geometrical sense a lot more complex. For example, sometimes face normal vectors do not point inside a solid geometry or faces are not properly connected. Apart from that, 3D city models are used in very specific applications with special quality requirements. For example, solar potential analyses highly depend on a model that is properly annotated with semantic information, but classification techniques can only succeed if the geometry does not contain contradictions. Apart from that, creating a physical model out of a virtual one using a 3D printer device does not work if the faces in the model are not connected correctly or—even worse—if they are not connected at all.

Due to this, work has been conducted to perform quality management on 3D geodata as well. For example, Kazar et al. describe methods for validating polygonal 3D geometries inside an Oracle database (Kazar et al., 2008). Nevertheless, this approach does not include healing capabilities. Krämer et al. concentrate on defining quality measurements for 3D city models (Krämer et al., 2007). Besides, this work includes simple algorithms for quality assessment and healing of geometries, although they are not very elaborated. More about automatic healing of 3D city models can be found in the paper by Bogdahn and Coors (Bogdahn and Coors, 2010).

In this paper we continue the work of Bogdahn and Coors and go one step further. We present the results of integrating validation and healing operations in a 3D geospatial server tailored to management and provisioning of urban models.

3 VALIDATION OF 3D CITY MODELS

In CityGML a building geometry is usually modelled as a set of Solids or as a MultiSurface. While a Solid represents the boundary of a rigid body, a MultiSurface is just a set of Polygons with no further constraint. According to the CityGML standard document, the volumetric parts of an object's geometry should be modelled using Solid elements, while the non-volumetric parts should be represented by MultiSurface elements (Gröger et al., 2008, p. 61).

The Linear Ring is the fundamental element to describe 3D geometries in CityGML. Every single polygon of a building geometry is defined by its boundary—the Linear Ring. In the following we define the geometry of a Linear Ring element.

A *sequence* is an ordered list of elements. Unlike a set, order matters, and the exact same elements can appear multiple times at different positions in the sequence. A finite sequence a with $n+1$ elements is denoted as $a = (a_0, a_1, \ldots, a_n)$. The empty sequence $a = ()$ has no elements.

A finite sequence of points $R = (P_0, P_1, \ldots, P_n), n \geq 3, P_i = (x_i, y_i, z_i)$ is a Linear Ring iff

(i) the first and last point P_0 and P_n represent the same point: $P_0 = P_n$ (*closeness*)

(ii) all points of the sequence besides start and end point are different:

$$\bigvee_{\substack{i=0\ldots n-1\\ k=0\ldots n-1\\ i\neq k}} P_i \neq P_k \tag{1}$$

(iii) two edges (P_i, P_{i+1}) and $(P_k, P_{k+1}), i = 0, \ldots, n-1, k = 0, \ldots, n-1, i \neq k$ do only intersect in one start-/endpoint. No other intersection is allowed (*no self intersection*).

If all points of the sequence are co-planar, the Linear Ring is *planar*.

The area enclosed by a planar Linear Ring (including the Linear Ring geometry) is a *Polygon*. The orientation of the polygon is given by the order of points within the sequence that defines the Linear Ring.

A *solid* geometry represents a rigid body. The surface of the solid is defined by a set of polygons with the following properties:

The set $C = \{S_1, S_2, \ldots, S_n\}$ of polygons bounds a solid iff:

(i) The intersection of two polygons S_k and S_l of C is either empty or contains only points P and edges e that are part of both Linear Rings. The polygon S_k is defined by the Linear Ring $R_k = (P_0^k, P_1^k, \ldots, P_n^k)$. The intersection of S_k and S_l equals:

$$S_l \cap S_k = \begin{cases} \emptyset \\ \{Q_0, Q_1, \ldots, Q_m\}, Q_j = P_i^k \\ \{e_0, e_1, \ldots, e_m\}, e_j = \overline{P_i^k P_{i+1}^k} \end{cases} \tag{2}$$

(ii) Every edge $e_k = \overline{P_i^k P_{i+1}^k}$ of a Linear Ring $R_k = (P_0^k, P_1^k, \ldots, P_n^k)$, defining a polygon $S \in C$, is used exactly once as an edge $e_i = \overline{P_j^l P_{j+1}^l}$ in a Linear Ring $R_l = (P_0^l, P_1^l, \ldots, P_m^l)$, defining another polygon $S_l \in C$ with $P_i^k = P_{j+1}^l$ and $P_{i+1}^k = P_j^l$.

(iii) All polygons in C are oriented such that the normal vector of each polygon points to the outside of the Solid.

(iv) All polygons in C are connected, that is the dual graph of C has a path containing all nodes. The dual graph $G_C = (V_C, E_C)$ of C consists of a set V_C of nodes and a set E_C of edges. Every node v of V_C represents exactly one polygon of C . An edge shared by two polygons S_k and S_l of C is represented by an edge $e = (v_{S_k}, v_{S_l})$ in E_C.

(v) For every point P of a Linear Ring of a polygon of C applies: The graph $G_P = (V_P, E_P)$, that is only built by polygons and edges that touch P, is connected. Each node v of V_P represents a polygon whose Linear Ring contains P. Two nodes are connected by an edge e of E_P iff the two polygons represented by the nodes have a common edge that touches P.[2]

It follows from (i) and (ii) that the surface defined by C has no holes. Together with conditions (iv) and (v) it follows that the inner of the solid defined by C is connected.

[2]The condition (v) is equivalent to the umbrella axiom presented by Gröger and Plümer (Gröger and Plümer, 2011). This axiom is defined as follows: each point is surrounded by exactly one cycle that is an alternating sequence of line segments and polygons.

3.1 Validation and Healing

In order to validate the geometry and topology of 3D city models we have developed a modular software framework that enables the user to define own check classes or pick some of the existing checks. Currently, checks are available for Linear Rings and Solids based on the above given definitions:

- C-LR-1: check that a Linear Ring consists of at least 4 points
- C-LR-2: check that a Linear Ring is closed
- C-LR-3: check that besides the first and the last point, each point appears only once in a Linear Ring
- C-LR-4: check that Linear Ring has no self-intersection
- C-LR-5: check that Linear Ring is planar

Figure 1: *Almost* planar linear ring

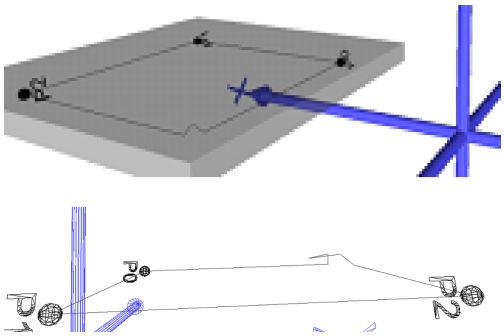

Figure 2: Bends and folds should not appear in planar Linear Rings

From practical point of view, Linear Rings that are not exactly planar should be accepted as well. Intuitively, a Linear Ring R would be defined as planar if there is a plane E with distance $d(E, P_i)$ less than a given threshold for all points P_i of R (see figure 1). However, this is true for Linear Rings with small bends and folds as shown in figure 2 as well. In order to avoid such situations, a slightly different definition of planarity of Linear Rings is introduced:

A Linear Ring R is planar iff it is a valid Linear Ring and the distance of all points to any plane E_{ijk} , that is defined by three co-linear points P_i, P_j and P_k is less than a given threshold:

$$\forall E_{ijk} \forall P_a = dist(P_a, E_{ijk}) \leq \varepsilon \quad (3)$$

We have specified three different algorithms for C-LR-5 to allow almost planar Linear Rings. The first algorithm takes three non-linear points from the Linear Ring and checks that the distance of the plane defined by these three points to all other points of the ring is less than a given threshold. It works well for most of the available models, but it allows bends and folds as well. To avoid this, a second algorithm was specified that takes into account all possible planes defined by any combination of three non-linear points of the Linear Ring. An alternative check is to triangulate the polygon defined by the Linear Ring and check the largest angle between the face normal of all pairs of triangles.

Apart from the checks for Linear Rings, we also defined checks for Solids:

- C-Solid-1: no self-intersection
- C-Solid-2: each edge bounds exactly two polygons
- C-Solid-3: face normal vectors point to the outside
- C-Solid-4: the dual graph of the solid boundary is connected
- C-Solid-5: for each vertex: the graph that is only build by polygons and edges that touch this vertex, is connected

Figure 3 shows an example of the dual graph of a given Solid. Each polygon in the Solid is represented by a node in the dual graph. Incident polygons are connected by an edge in the dual graph. C-Solid-4 validates whether the dual graph is connected.

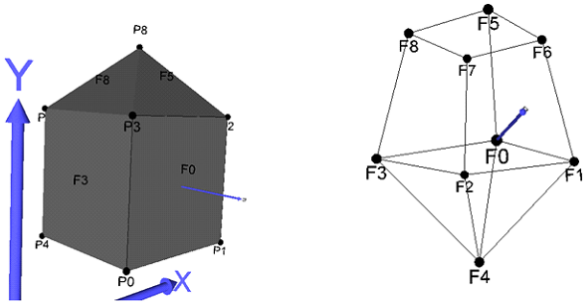

Figure 3: The dual graph of the boundary of a solid

If one of the checks fails, we try to heal the model. This is currently not possible for all checks. For Linear Ring elements, healing is applied for C-LR-2 by inserting a closing point, for C-LR-3 by removing redundant points, and for C-LR-4 by inserting additional points in case of point intersection. After healing, the Linear Ring can still contain errors, so it is validated again. If C-LR-1 or C-LR-5 fail, the model cannot be healed so far. For Solid elements, healing is applied for C-Solid-3 (flip orientation) and for C-Solid-2 if there is a Linear Ring of edges that bounds only one polygon—this is a hole that can be closed.

4 INTEGRATION INTO THE CITYSERVER3D

The quality assurance framework is currently provided as a software library that can be used by different geospatial applications. As a first step, it has been integrated into the geospatial management application CityServer3D[3] developed by the Fraunhofer IGD. This product consists of a server component that saves geospatial data in an object-relational database as well as several client applications that can be used to edit and visualize the data. Apart from that, the application supports several standardised data exchange formats like CityGML, KML, X3D as well as OGC service interfaces like W3DS.

[3] http://www.cityserver3d.de/en/

To control the test parameters the CityServer3D's client UI has been extended. The user is able to select the different test algorithms, execute them and then review the results. The errors found and the proposed changes are displayed in a table view. By clicking an entry in this table, a 3D visualisation will be presented, comparing the original object with the fixed one. The user can then decide if he likes to apply the proposed fix or not. After that, changes can be written to the database or exported as a file in the CityGML format, for example. The test results will be saved in an ISO 19115 conformant way. Therefore metadata will be attached to evaluated objects.

Internally, the CityServer3D uses a common data model to hold the spatial objects in memory and to manage them in the database. This allows processing the data in a unified way. In order to use the quality assurance library, the data has to be converted. The CityServer3D uses a surface-based approach to describe geometries, so conversion can be reduced to simple schema mapping. When fixes are applied, the corrected objects have to be converted back to the CityServer3D's common data model. Special care has been taken to avoid information loss during the conversion—especially regarding semantic information like topology and metadata.

Figure 4 shows how the quality assurance library integrates into the architecture of the CityServer3D. The library can be installed as a plug-in into the CityServer3D main client product—the so-called AdminTool. It is controlled by a special UI component that integrates tightly into the existing interface. Apart from that, the quality assurance plug-in has direct access to the 3D city model currently loaded into the AdminTool.[4] A thin converter maps objects from the CityServer3D's internal data model to ones that can be managed by the quality assurance library and vice versa.

5 RESULTS

The validation algorithms have been tested with a 3D city model consisting of 2 650 building objects with LoD2 geometries. Some buildings consisted of several unconnected building parts. In total, 47 426 polygons were used, 3 326 polygons for ground surfaces, 38 732 polygons for wall surfaces, and 5 368 polygons for roof surfaces. Figure 5 shows the distribution of numbers of polygons per building. Figure 6 shows the results of validation and the success of automatic healing of the models.

The images show that the quality assurance library is already able to find a lot of geometrical defects in a real-world 3D city model. Apart from that, healing defects is also already possible for most of the identified buildings.

The development of the library as a closed system which can be used stand-alone or as an integrated component in other applications appears to be a sensible approach. Many existing applications with focus on 3D data management or manipulation can benefit from quality test capabilities. Integrated quality test functionality can also optimize workflows based on existing applications and increase acceptance among users.

The modular architecture of the CityServer3D allows adding new features through plug-ins. Integrating the library into the CityServer3D went quite well as it had only to be wrapped into a simple plug-in. Apart from that, a converter had to be implemented mapping the 3D city model from the CityServer3D's internal data model to objects understood by the library and vice versa. Although a dedicated user interface for the test algorithms already existed, we decided to extend the existing UI of the CityServer3D's AdminTool in order to allow a seamless integration.

[4]The city model can be imported from either a file or from the database.

ACKNOWLEDGEMENTS

The authors would like to thank Gerd Gröger and the SIG-3D quality working group for fruitful discussion and the German Ministry for Education and Research (BMBF) for funding the project CityDoctor.

REFERENCES

Bogdahn, J. and Coors, V., 2010. Towards an automated healing of 3D urban models. Kolbe, T., König, G. and Nagel, C. (Eds.); Proceedings of International Conference on 3D Geoinformation, Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Science XXXVIII-4/W15, Shaker Verlag GmbH, Aachen, pp. 13–17.

Gröger, G. and Plümer, L., 2011. How to achieve consistency for 3D city models. Geoinformatica(15/1), Elsevier pp. 137–165.

Gröger, G., Kolbe, T., Czerwinski, A. and Nagel, C., 2008. OpenGIS City Geography Markup Language (CityGML) Encoding Standard, OGC reference number OGC 08-007r1 version 1.0.0.

Guptill, S. C. and Morrison, J. L., 1995. Elements of spatial quality. Elsevier Science, Kidlington, Tarrytown, Tokyo.

Kazar, B. M., Kothuri, R., van Oosterom, P. and Ravada, S., 2008. On valid and invalid three-dimensional geometries. Fendel, Oosterom, Penninga, Zlatanova (Eds.); Advances in 3D Geo Information Systems, Springer Press.

Krämer, M., Haist, J. and Reitz, T., 2007. Methods for spatial data quality of 3D city models. De Amicis, R. and Conti G. (Eds.); European Association for Computer Graphics (Eurographics): Eurographics Italian Chapter Proceedings pp. 167–172.

MacEachren, A., Robinson, A., Hopper, S., Gardner, S., Murray, R., Gahegan, M. and Hetzler, B., 2005. Visualizing geospatial information uncertainty: What we know and what we need to know. Cartography and Geographic Information Science 21(3), pp. 139–160.

TC 176/SC, 2005. ISO 9000:2005, Quality management systems–Fundamentals and vocabulary. International Organization for Standardization.

Van Oort, P., 2005. Spatial data quality: from description to application. Optima Grafische Communicatie, Rotterdam, The Netherlands.

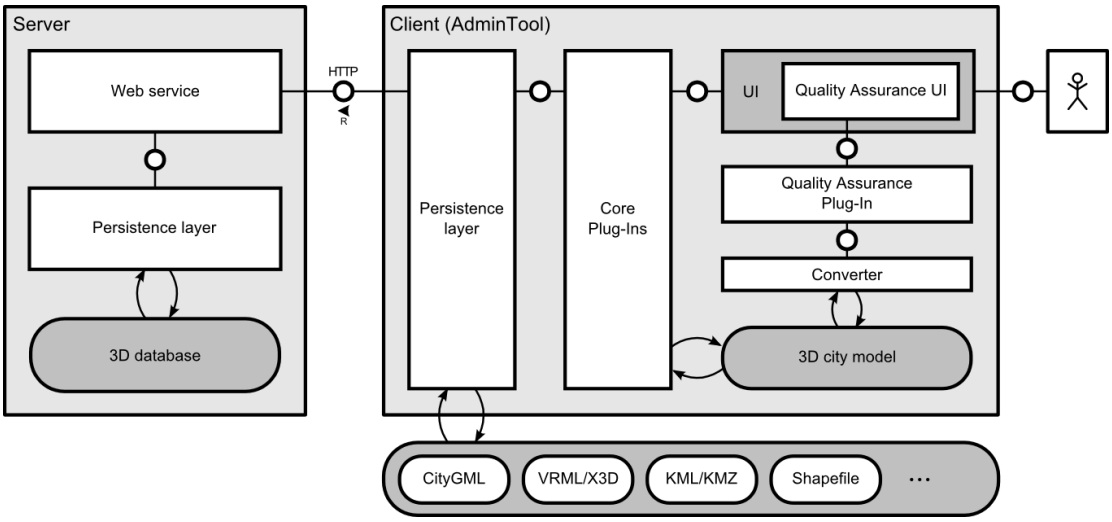

Figure 4: Architectural diagram showing the integration of the quality assurance component into the CityServer3D

	Total	per building
Buildings	2650	
Polygons LOD1	0	
Polygons LOD2	47426	17,90
Ground LOD2	3326	1,26
Wall LOD2	38732	14,62
Roof LOD2	5368	2,03
unconnected components	3134	1,18

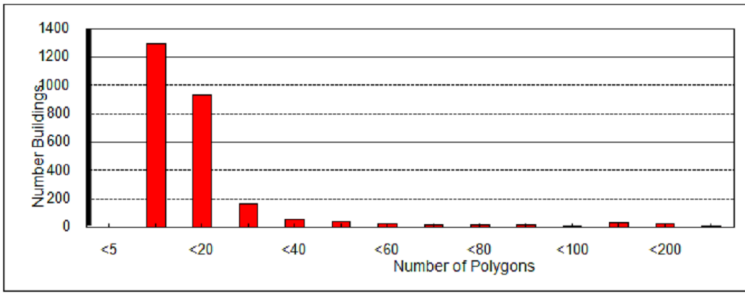

Figure 5: The test data set contains 2 650 buildings (LoD2 geometry) with 47 426 polygons in total. The diagram shows the distribution of polygons per building

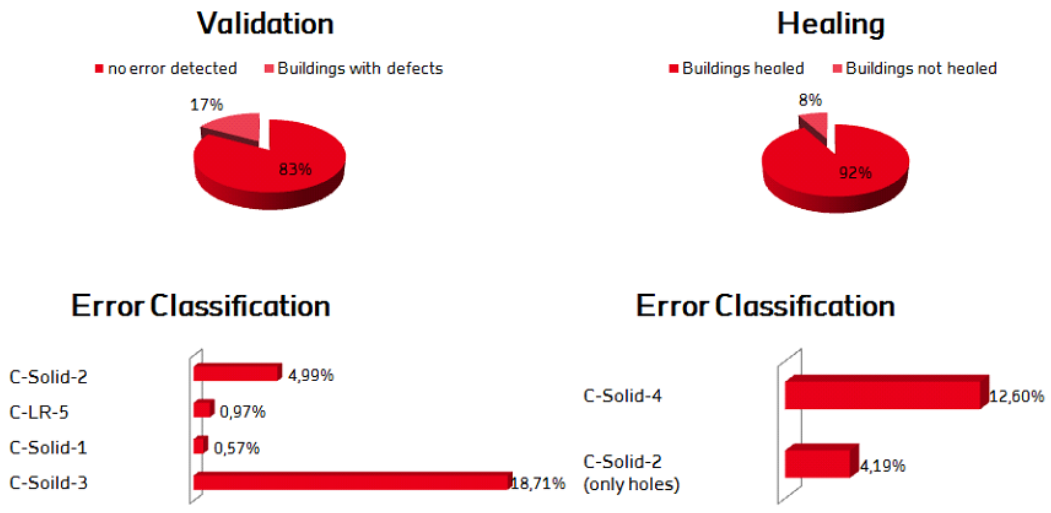

Figure 6: Validation and healing results