Geometric-Semantical Consistency Validation of CityGML Models

Detlev Wagner, Mark Wewetzer, Jürgen Bogdahn, Nazmul Alam, Margitta Pries and Volker Coors

Abstract In many domains, data quality is recognized as a key factor for successful business and quality management is a mandatory process in the production chain. Automated domain-specific tools are widely used for validation of business-critical data. Although the workflow for 3D city models is well-established from data acquisition to processing, analysis and visualization, quality management is not yet a standard during this workflow. Erroneous results and application defects are among the consequences of processing data with unclear specification. We show that this problem persists even if data are standard compliant and develop systematic rules for the validation of geometric-semantical consistency. A test implementation of the rule set and validation results of real-world city models are presented to demonstrate the potential of the approach.

D. Wagner (✉) · J. Bogdahn · N. Alam · V. Coors
HFT Stuttgart—University of Applied Sciences, Faculty C, Schellingstraße 24, 70174, Stuttgart, Germany
e-mail: detlev.wagner@hft-stuttgart.de

J. Bogdahn
e-mail: juergen.bogdahn@hft-stuttgart.de

N. Alam
e-mail: nazmul.alam@hft-stuttgart.de

V. Coors
e-mail: volker.coors@hft-stuttgart.de

M. Wewetzer · M. Pries
Beuth Hochschule für Technik Berlin—University of Applied Sciences, Department II, Luxemburger Straße 10, 13353, Berlin, Germany
e-mail: mark.wewetzer@bht-berlin.de

M. Pries
e-mail: margitta.pries@bht-berlin.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_10,

1 Introduction

A steadily growing number of application fields for large 3D city models, such as navigation or solar potential analysis, have emerged in recent years. All of them are strongly relying on the quality of the underlying models. Besides semantic, topological or visualization aspects, geometric correctness is a major factor for the quality of virtual city models. So far, common standards defining correct geometric modeling are not precise enough to define a sound base for data validation. Depending on the application domain, different standards and guidelines allow various modeling alternatives. This fact causes difficulties for data suppliers as well as for customers due to the need of explicitly stating detailed requirements.

Furthermore, modeling and exchange formats like CityGML (Gröger et al. 2008) do not include formal constraints with respect to geometric-semantical consistency.

In this document we describe the validation of virtual urban models, mainly focused on CityGML format. We define basic checks to be performed on a given data set, where geometric requirements should be tested and validated to assure a user-defined level of quality. This approach focuses on common errors which are typically found in 3D city models. A prototype of these checks working with CityGML data sets is implemented in a system developed at HFT Stuttgart.

In the next step, validated geometry has to be checked for semantic consistency and plausibility. A concept built on a formal set of strict rules is presented.

2 Relevance of Quality Management Of Three-Dimensional Models

Data validation is the process of checking data against a predefined set of validation rules. The main goal is to ensure the correctness, integrity and consistency of the data set regarding certain requirements, which is *influenced by the field of application*. A second benefit is that validated data is standard compliant (if the standard is considered by the rule set) and has advantages with respect to interoperability on different systems. Data exchange should be based on validated data since both data supplier and customer have agreed on common underlying rules.

2.1 Quality Assurance of 3D models in Other Domains

It is acknowledged in the geoinformation domain that "the representation of 3D objects using CAD (Computer Aided Design/Drafting) is not new, and significant work has been done on ensuring that the computer-based model is valid" (Thompson and Oosterom 2011). It seems attractive to figure out if existing

methods from the CAD, reverse engineering or computer visualization domain would be suitable for geodata as well, because quality issues have gained great importance since many years.

Some examples: Data transformation from one CAD-system to another could lead to "dirty geometry" caused by different tolerance values (Wöhler et al. 2009); if data for reverse engineering purposes is collected with a range scanner, it can be noisy or incomplete, because the scanner was not able to capture every wrinkle of the model correctly (Rocchini et al. 2004).

Quality issues of CAD-models have lead to the formulation of ISO standard ISO/PAS 26183 (International Organization for Standardization (ISO) 2006), also well known as SASIG PDQ[1] Guideline v2.1, defining and categorizing common errors, especially in the automotive industry. A lot of commercial software for checking and healing geometry has been developed, e.g. CADfix (International TechneGroup 2011), CADdoctor (Elysium 2008), VALIDAT (T-Systems International 2011) or Q-Checker (Transcat PLM 2011).

Quality related research of polygonal models is ongoing since more than 10 years for computational applications. Often these are restricted to surface models consisting of triangles, thus most of the healing algorithms are limited to triangles (Ju 2009). Because surface modeling in CAD applications is more error-prone than polygon based solid modeling, research was focused on healing free-form surfaces rather than polygons. Additionally, repairing complex models is not trivial and even today it is normally not possible to heal a corrupted model fully automatically (Wöhler et al. 2009; Butlin and Stops 1996).

Nonetheless the discussed problems are similar to those occurring in 3D city models: (Borodin et al. 2002) comment on the finding and filling of holes in surface models, or resolving self intersections. The latter topic is also discussed by Attene and Falcidieno (2006), Yamakawa and Shimada (2009) and Rocchini et al. (2004). For a broader overview we recommend the paper of (Ju 2009).

The efforts and achievements made so far are a good guidance for the development of similar documents and tools for 3D city model.

2.2 Validation of Geodata and 3D City Models

Although the advantages of assessing and managing spatial data quality are widely recognized, their application is not yet widespread (van Oort 2005). Today, the transition from traditional 2D mapping towards 3D modeling is in full progress for many real-world applications, and high data quality is becoming essential for applications such as cadastre or urban planning. Efforts to translate validation concepts from the 2D GIS world to 3D data are one of the approaches, eventually trying to link the two domains (Ghawana and Zlatanova 2010).

[1] Strategic Automotive Data Standards Industry Group—Product Data Quality.

ISO standards referring to spatial data quality exist (International Organization for Standardization (ISO) 2002; International Organization for Standardization (ISO) 2003a; International Organization for Standardization (ISO) 2003b), other standards for 3D data are well established, e.g. CityGML (Gröger et al. 2008). The OGC domain working group for data quality mentions a number of categories for quality measures, including accuracy, completeness, consistency and definition for semantic interoperability (Open Geospatial Consortium (OGC) 2011). However, *there is no generally acknowledged definition of quality so far.* As stated by (Bogdahn and Coors 2010), *quality aspects for 3D city models are highly dependent on user preferences and the field of application.* A comprehensive overview of geometry validation is described by Kazar et al. (2008) and implemented in an Oracle Spatial database system.

A series of vendor specific implementations in commercial software packages for the validation of 3D geometries came to the market in recent years. For example, Oracle integrated validation rules and algorithms for 3D geometries in their database system Oracle Spatial (Kazar et al. 2008), and ESRI's wide-spread shapefiles were extended to handle 3D geometries. As internal data models and processing rules are incompatible, interoperability becomes a challenge if not impossible. Thus, it became necessary to define the structure of the data in more detail, e.g. for geometry (Oosterom et al. 2005) or topological consistency and analysis (Ledoux 2011; Boguslawski et al. 2011). Specific rules and constraints based on user needs are developed to extend existing standards (Gröger and Coors 2010).

An overview of validation concepts and needs is presented by Karki et al. (2010). Their implementation is based on the decomposition of solids into tetrahedrons for validation of a 3D cadastre and describes entry-level validation rules in terms of continuity, reasonability and spatio-temporal aspects. All of the above-mentioned approaches are dealing with geometry or topology, not considering semantic information.

Based on the semantic model of CityGML (Stadler and Kolbe 2007) distinguish six categories of model complexity and structure wrt. spatio-semantical coherence. Only models where semantic components correlate to geometric components on the same level of hierarchy can be fully coherent. Thus, "it becomes clear that besides the spatial and semantic complexity also the coherence of the two structures is an important quality aspect of 3D city models" (ibid.). CityGML support coherent representation of geometry and semantics. (Métral et al. 2009) underline the relevance of semantically enriched models in their ontology-based approach.

Currently, discussions in the German quality working group of SIG-3D[2] on correct usage of attributes and modeling alternatives are ongoing. The goal of the working group is to suggest "best-practice" rules for 3D city modeling. A handbook with modeling guidelines for typical elements is in preparation. Partially, these guidelines are directly derived from definitions in the standard. In

[2] SIG-3D—GDI-DE; Special Interest Group 3D of the Geo Data Infrastructure Germany.

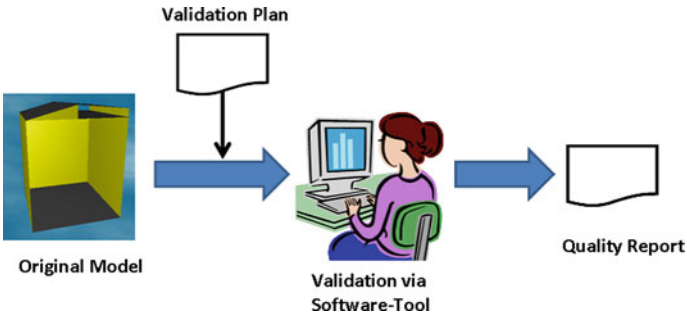


Fig. 1 Quality check process

many cases, however, the standard allows several alternatives. For example, the basic element `Building` can be modeled as `Solid` or as `MultiSurface` geometry. Some attributes might refer to different features without being clearly defined. A prominent example in this context is `MeasuredHeight` which reflects the building height which was measured by surveying methods. However, it is not clear if this value refers to the difference between ground surface and highest point of the structure, highest point of the roof, average roof height, etc.

Beyond the mere definition of these elements and attributes is their relation to the geometry. Intuitively, the geometric model height should be less or equal than the given attribute value of `MeasuredHeight`. Although attributes with geometric relevance encompass many obvious restrictions, geometric-semantical consistency is neglected so far. A reason might be the lack of recognized rules and software implementations.

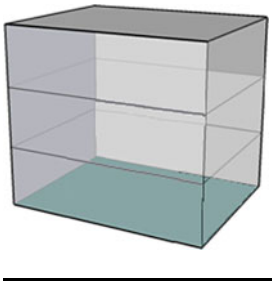
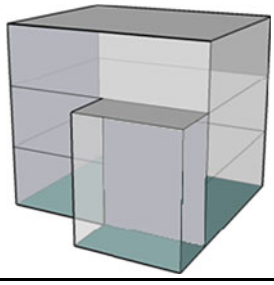
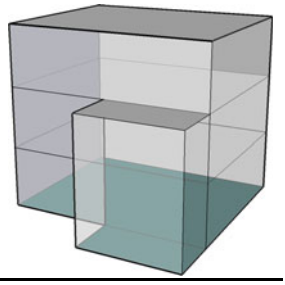
In this paper, we present a concept to validate the consistency of geometry and semantics of CityGML models and define adequate rules and restrictions.

3 Concept of Quality Assurance for 3D City Models

A first concept of quality management for urban 3D models is presented by Coors and Krämer (2011). In this paper the implementation of a prototype for validation and healing of geometry errors is described (Fig. 1).

Gröger and Plümer (2009) developed a set of axioms to achieve geometric-topological consistency of 3D models. (Gröger and Coors 2010) defined valid geometry elements for CityGML in a more detailed approach. They add useful restrictions which are based on modeling guidelines discussed within the quality working group (SIG-3D Quality Working Group 2012). These extended rules are the base for the geometric model which we use for validation of CityGML data. The rules are applicable for data stored in XML documents as well as in data base solutions, which are customized adequately (Pelagatti et al. 2009).

Table 1 Modeling alternatives for the CityGML element `Building`

Alternative 1:
`1.Building (Solid)`
	Alternative 2:
	`1 Building (Solid)`
	`1 BuildingPart (Solid)`
	`combined geometry:`
	`CompositeSolid`
		Alternative 3:
		`1 Building`
		`(MultiSurface)`
		`1 BuildingPart`
		`(MultiSurface)`
		`combined geometry:`
		`Solid`

CityGML allows many different alternatives for modeling. This is an obstacle in the validation process, because it is not unambiguously defined what validity actually means without further specification. A verbal description of the modeling rules is a first step to formulate constraints for geometry, topology and semantics, but cannot be used for automated validation tools.

Simple examples are the CityGML elements `Building` and `BuildingPart`, which can be modeled in at least three different ways (Table 1):

- as single `Solid`
- as `CompositeSolid` (with XREF for shared faces)
- as `MultiSurface` geometry

All three alternatives are valid in CityGML. Validation of the models is only possible if the validation rules include information on the underlying modeling principles. Then `Buildings` which are modeled according to different rules can be detected as errors. It should be possible to allow several alternatives at the same time.

To achieve geometric-semantical consistent models it is necessary to

1. specify allowed alternatives with formal rules, and
2. validate the geometry.

3.1 What is Valid Geometry?

Validation against the schema definition of CityGML is a basic requirement to ensure a standard compliant XML structure. However, no assertion on the geometric correctness of modeled objects is possible: A building can be described by a

syntactically valid CityGML document but still have geometric errors like holes or non-planar faces, or contain inconsistent attribute values.

The following definitions are used in this paper. We stick quite close to the GML standard and limited the definitions to CityGML. Normally a polygon is bounded by a `gml:Ring`, but as there is only the linear ring as subtype of `gml:Ring` in CityGML we refer directly to `gml:LinearRing`.

3.1.1 Definitions of geometric elements

1. Point

   The element `gml:Point` is given as a triple of three real numbers:

$$p := (x, y, z), x, y, z \in \mathbb{R}$$

2. Edge

   A directed edge e is defined by an ordered pair of two points:

$$e_{k,l} := (p_k, p_l)$$

3. Linear Ring

   A sequence $R := (p_0, p_1 \ldots, p_n)$ of points p_i, with linear interpolation between the points (a sequence of connected edges) is called a `gml:LinearRing`, if $p_0 = p_n$ and $n \geq 3$. Note that the direction of an edge in a linear ring is determined by the order of the points.

4. Polygon

   By definition, a `gml:Polygon` P is a surface bounded by a `gml:LinearRing`. The boundary is coplanar and the polygon uses planar interpolation in its interior. Hence, a linear ring is planar if all points of the sequence are coplanar.

5. Solid

   A `gml:Solid` S is defined by a set $S := (P_1, \ldots, P_m)$ of connected polygons $P_i, i = 1, \ldots, m$. The set of polygons defines a 2-manifold surface.

3.1.2 Validity Axioms

It is quite clear, that the pure definitions of entities derived from the GML standard could lead to non-manifold, but still valid geometry. To avoid degenerated geometry we define additional validity axioms which the geometry has to satisfy to be valid in accordance with Gröger and Plümer (2009).

Linear Ring and Polygon

Let $\mathrm{R} = (\mathrm{p}_0, \mathrm{p}_1 \ldots, \mathrm{p}_n)$ be a linear ring and let P be the polygon defined by R. R and P are called valid, if

a. R consists of at least four ordered points (CP-NUMPOINTS);
b. The first and last point are identical: $p_0 = p_n$ (CP-CLOSE);
c. All points of the sequence besides the first and the last are different: $p_i \neq p_j,\ i,j = 0,\ldots n-1,\ i \neq j$ CP-DUPPOINT);
d. Two edges $e_{i,i+1}$ and $e_{j,j+1},\ \ i,j = 0,\ldots n-1,\ i \neq j$ of R do only intersect in one start-/endpoint. No other intersection is allowed (no self-intersection, CP-SELFINT);
e. According to definition (4), the boundary of P has to be planar (CP-PLAN-DIST, CP-PLANDISTALL, CP-PLANTRI).

Solid

Let $S = (P_1, \ldots, P_m)$ be a solid. S is called valid, iff

f. all *polygons* of S are valid (axioms a to e);
g. S *consists* of at least four polygons (CS-NUMFACES);
h. each edge e of a polygon of S is exactly referenced once by another polygon of S (CS-2POLYPEREDGE);
i. all *polygons* in S are oriented such that the normal vector of each polygon points to the outside of the solid. This means that an edge $e_{v,w}$ of a polygon P_k is directed in opposite direction in polygon P_l, which is sharing $e_{v,w}$: $e_{v,w} \in P_k \Rightarrow e_{w,v} \in P_l$ (CS-FACEORIENT, CS-FACEOUT);
j. the intersection of two polygonsP_k andP_l of S is either empty or contains only points p or edges e (points included), that are part of both polygons:

$$P_k \cap P_l = \left\{ \begin{array}{c} \emptyset \\ \{p | p = (p_0, \ldots, p_c), \quad p_i \in P_k \wedge p_i \in P_l\} \\ \left\{ \begin{array}{c} e | e = \left(e_{v_0,w_0}, \ldots, e_{v_d,w_d}\right), \\ \left(e_{v_i,w_i} \in P_k \wedge e_{w_i v_i} \in P_l\right) \wedge \\ \left(p_{v_i}, p_{w_i} \in P_k \wedge p_{v_i}, p_{w_i} \in P_l\right) \end{array} \right\} \end{array} \right.$$

(CS-SELFINT);

k. all polygons in S are connected (CS-CONCOMP);
l. each point is surrounded by exactly one cycle that is an alternating sequence of line segments and polygons, also called umbrella axiom (CS-UMBRELLA).

Based on these definitions geometry validation can be limited to the validity axioms of a solid for `Solid` geometries and the validity axioms of polygons for a `MultiSurface` geometry.

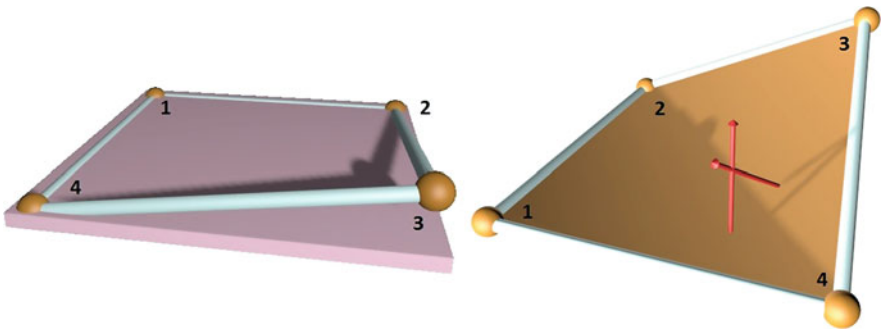


Fig. 2 Two examples of a non-planar `LinearRing`

3.2 Description of Check Routines

Five major check routines can be derived from the polygon definition above. The planarity checks are described in detail below. The other checks and the solid checks are defined in Coors and Krämer (2011). The relation of the axioms and the Check-IDs are given in Table 4.

Planarity can be defined in different ways as an internal discussion in the German Quality Working Group has shown and is referred to in Gröger and Coors (2010). Our implementation includes three different algorithms as a consequence of this discussion process, a fourth one is not yet implemented.

- Algorithm 1 (CP-PLANDIST)

Three non-linear points p_0, p_1, p_x of a linear ring R with $n + 1$ points describe a plane E with normal vector n. All other points of R must be situated in E (small deviations $\varepsilon \in \mathbb{R}$ are allowed):

$$|(p_0 - p_i) \cdot \boldsymbol{n}| < \varepsilon, i \in \{0, \ldots, n-1\}$$

In `LinearRing` $R = \{4, 1, 2, 3, 4\}$ depicted in Fig. 2 (left), three points (4,1,2) would define a plane (visualized in gray), but Point 3 is not located in it. This polygon can still be considered planar if the distance d between Point 3 and the plane is less than a preset maximum deviation ε. Folds and sharp bends cannot be detected reliably, however.

- Algorithm 2 (CP-PLANDISTALL)

It tests if all points of a linear ring R with $n + 1$ points are situated in all possible planes E_l with normal vectors n_l, defined by three non-linear points p_i, p_j, p_k of R (small deviations $\varepsilon \in \mathbb{R}$ are allowed):

$$|(p_i - p_a) \cdot n_l| < \varepsilon, a \in \{1, \ldots, n-1\}$$

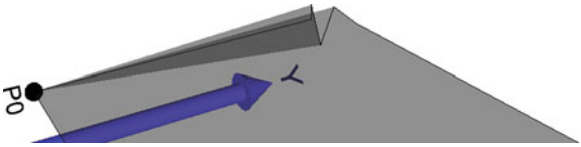


Fig. 3 `Linear Ring` with a fold

In contrast to CP-PLANDIST, this algorithm detects folds and sharp bends such as depicted in Fig. 3.

- Algorithm 3 (CP-PLANTRI)

Consider T as a triangulation of the polygon defined by R, containing m triangles. Additionally, let $n_l, l \in \{0, \ldots, m-1\}$ be the unit normal vector of the *l-th* triangle in T. R is planar, iff each scalar product of unit normal vectors n_a and n_b of two different triangles of T is less than a tolerance value ε:

$$\boldsymbol{n}_a \cdot \boldsymbol{n}_b < \varepsilon, a \neq b; \quad a, b \in \{1, \ldots, m-1\}$$

Folds and sharp bends are detected. Fewer problems with long polygons occur because the normal vector does not change its direction whereas point distance to a reference plane can increase compared to smaller polygons.

- Algorithm 4 (CP-PLANADJUST, not yet implemented in our validation tool, cf. Sect. 4)

An adjustment plane E with normal vector n for all points p_i of a linear ring is computed. All points p_i must be situated in E (small deviations $\varepsilon \in \mathbb{R}$ are allowed):

$$|(p_0 - p_i) \cdot n| < \varepsilon$$

However, as with CP-PLANDIST, folds and sharp bends cannot be detected reliably by CP_PLANADJUST.

Results of the error distribution of test models are given in Table 4. The highest number of non-planar polygons is detected by CP-PLANTRI, the lowest number is found by CP-PLANDIST. CP-PLANDISTALL is supposed to find all possible planarity errors as well; hence it is surprising that the number is different from the CP-PLANTRI result in some models. In both cases the tolerance value was set to 0.01. This reflects the maximum point distance for CP-PLANDISTALL and the maximum angle for CP-PLANTRI. Obviously, the thresholds are not comparable and CP-PLANTRI is more sensitive.

Another issue is that polygons with a long and narrow shape are expected to have a large deviation for certain point combinations in CP-PLANDISTALL. This leads to the question how to determine a useful value for the tolerance value ε. Our implementation is using a default value of 0.01, which can be changed by the user.

Table 2 Dependencies of geometry check routines

	CP-NUM-POINTS	CP-DUP-POINT	CP-PLAN[a]	CS-2POLY-PEREDGE	CS-FACE-ORIENT
CP-SELFINT	–	•	–	–	–
CP-PLAN[a]	•	•	–	–	–
CS-SELFINT	–	–	•	–	–
CS-FACEORIENT	–	–	–	•	–
CS-FACEOUT	–	–	–	–	•
CS-UMBRELLA	–	–	–	•	–

[a] CP-PLAN means either CP-PLANDIST, CP-PLANDISTALL or CP-PLANTRI

3.3 Dependencies in the Check Process

Some checks perform correctly only if the input has passed other checks before processing. This might be necessary to guarantee that certain assumptions are valid. For example, it would not make sense to test a polygon for planarity if there are less than 4 points, because planarity is not defined in that case.

In a strictly designed check process, all checks (except basic CP-NUMPOINTS) should depend on each other in a nearly linear way, according to the order as they are listed in Sect. 3.2. During testing we noticed that such strong restrictions prevent parts of the geometry to be checked by higher-order checks in case basic checks have not been passed. *The check process stops for the whole building geometry, even if there is only a CP-DUPPOINT error in one polygon. As a consequence, some checks at the end of the dependency chain are not executed for some buildings.*

To decrease the number of unchecked buildings we decoupled some checks from their former dependency. Table 2 gives a quick overview on all remaining dependencies which are underlying the current check process of our implementation. For example, CS-FACEOUT yields correct results only for solids which have passed CS-FACEORIENT successfully.

All other checks work correctly for geometric input extracted from CityGML files which can be validated against the current XML schema definition.

3.4 Concept of Semantic Validation

CityGML defines a semantic model that enables the user to add further information to spatial object. This goes far beyond the pure geometrical description of features. The idea is to enable a model to “know” what kind of features it contains and not only their location and shape. It can be enriched with domain-specific information which exceeds the default specification of CityGML. However, semantic information can be inconsistent with the geometry, e.g. if a building wall is defined as a `RoofSurface`.

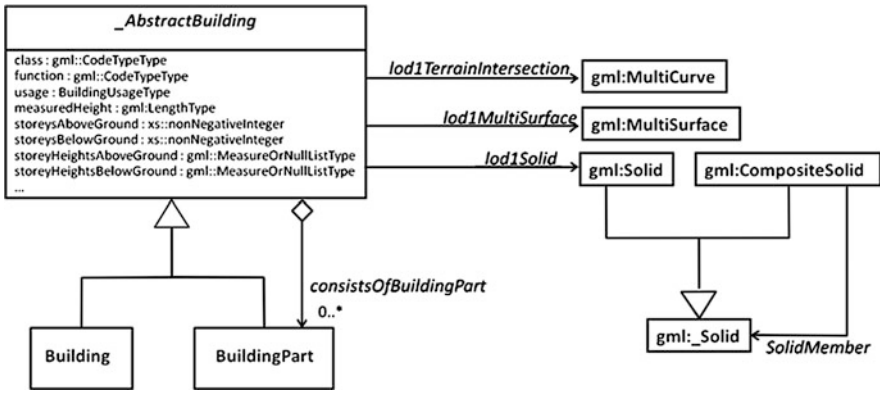


Fig. 4 UML diagram for CityGML element *_AbstractBuilding*

Based on validation of geometrical features as explained above, we introduce a concept for the validation of geometric-semantic consistency. Valid geometry is the base before other geometry-related information can be validated. A set of applicable and unambiguous rules is defined, mainly determined by the underlying data model (Stadler and Kolbe 2007).

For a start, the concept is focused on the element `Building` as specified in CityGML 1.1 standard (Gröger et al. 2008). A `Building` can have one geometry per level of detail (LoD) in CityGML. These geometries will be treated separately. In this paper, we limit the discussion to LoD 1, where the geometric-semantic themes volume part of the building shell (`gml:SolidType`), surface part of the building shell (`gml:MultiSurfaceType`), terrain intersection curve (`gml:MultiCurveType`) and building parts (`BuildingPartType`) are defined. The relevant part of the UML diagram is shown in Fig. 4.

3.5 Constraints for Building Models in CityGML

Geometry of LoD1 is either modeled as `MultiSurface` or as `Solid` and `CompositeSolid` respectively. A terrain intersection curve to define the intersection line of the building geometry with the terrain model may exist in addition.

A building can be modeled in several parts (`BuildingPart` elements), representing individual structural elements. The CityGML standard defines base requirements for conformance of `Building` elements:

"If a building only consists of one (homogeneous) part, it shall be represented by the element `Building`. However, if a building is composed of individual structural segments, it shall be modeled as a `Building` element having one or more additional `BuildingPart` elements. Only the geometry and non-spatial

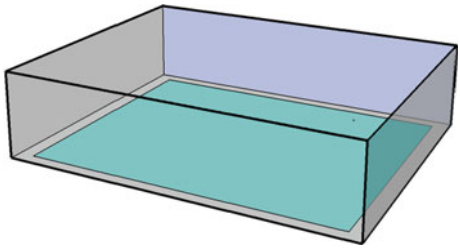


Fig. 5 Simple extrusion solid

properties of the main part of the building should be represented within the aggregating `Building` element" (Gröger et al. 2008).

Besides basic requirements no detailed rules are given in the standard how to model a `Building` correctly in a standardized form. Although there are comments or recommendations on the separation into several `BuildingPart` elements, different valid alternatives are possible according to the specification. However, a consistent modeling concept should be ensured throughout the actual city model, which is not included in a standard CityGML document. Hence, validation requires the formulation of rules in a standardized and machine-readable format.

3.6 Modeling Guidelines and Recommendations

In the simplest case a building geometry in LoD1 consists of an extrusion solid (Fig. 5). It should always be modeled as a `Solid`. The standard allows a `MultiSurface` in theory as well, which is widely used in practice. However, this way of modeling is not very helpful considering the consequences for potential applications of the model (e.g. watertightness).

3.7 Rules for Validation

In the following paragraphs a validation rule set is developed step by step, to ensure that a model is built of consistent geometric and semantic elements. The rules are given in colloquial language as well as in Object Constraint Language (OCL) as machine-readable format that can be processed automatically (Object Management Group 2011). The OCL statements given below are referring to the UML diagrams which define the CityGML elements in the standard. The relation of buildings and their parts is shown in Fig. 4, which includes the geometric elements for a valid model as well as the different alternatives for solids.

Additional rules are required to enable strict validation of models. We recommend a rule set on the basis of the modeling guidelines of SIG-3D (SIG-3D

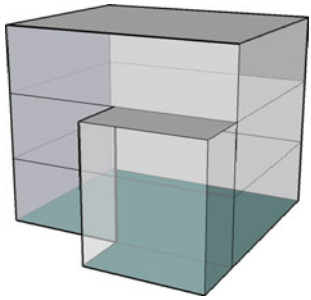


Fig. 6 Simple LOD 1 building with several structural elements

Quality Working Group 2012). Other alternatives could allow the users to deviate from these standards deliberately in case they require customized rules.

3.7.1 Simple Building Without Building Parts

Rule 1: If a building has no separable structural elements it is modeled as a simple `Solid` in LoD1:

```
context Building
inv: self.oclIsTypeOf(t:Building)
inv: self.lod1Solid->notEmpty() and
self.lod1MultiSurface->isEmpty()
inv: self.consistsOfBuildingPart->isEmpty()
```

3.7.2 Simple Building with Building Part

Different alternatives for modeling have to be considered in case a building is composed of several structural elements. This is illustrated in the following example building with different heights (Fig. 6).

This building can be modeled as a `Building` according to Rule 1, and no `BuildingPart` element is necessary in this case.

Just as well it is possible to model the building as `Building` with additional `BuildingPart`. The `BuildingPart` contains the lower structural segment at the front side of the building in Fig. 6. Different alternatives are considered subsequently:

Rule 2: If a building consists of several parts the main element is modeled as a `Solid`, as well as all other building parts. The combined geometry of all parts forms a `CompositeSolid`:

```
context Building
inv: self.oclIsTypeOf(t:Building)
inv: self.lod1Solid->notEmpty()and
self.lod1MultiSurface->isEmpty()
inv: self.consistsOfBuildingPart->notEmpty()->
```

```
forAll(b:BuildingPart | b.lod1Solid->notEmpty()
and b.lod1multisurface->isEmpty())
inv: self.lod1Solid->union(union
(self.consistsOfBuildingPart.lod1Solid))
= compositeSolid
```

The restriction to a CompositeSolid includes the requirement that all `Building` and `BuildingPart` elements are connected as stipulated by the standard.

The geometry according to rule 2 contains one or more common planes between adjacent solids (compare Table 1, alternative 2). This can be avoided by modeling the individual geometries as `MultiSurface` elements instead of `Solid` elements.

Rule 3: If a building consists of several parts the main segment is modeled as a `MultiSurface` element, as well as all other parts. The complete geometry of all building parts forms a `Solid`, i.e. the geometries are connected and the aggregated geometry could be defined by a single `Solid`:

```
context Building
inv: self.oclIsTypeOf(t:Building)
inv: self.lod1Solid- > isEmpty() and
self.lod1MultiSurface-> notEmpty()
inv: self.consistsOfBuildingPart- > notEmpty()->
forAll(b:BuildingPart | b.lod1Solid- > isEmpty()
and b.lod1multisurface- > notEmpty())
inv: self.lod1MultiSurface
- > union(union(self.consistsOfBuildingPart- >
 any(exists(lod1MultiSurface))).isTypeOf(lod1Solid)
```

We recommend modeling according to rules 1 and 2. The rules can by combined: rule 2 OR rule 3 would allow using both alternatives in one model.

A `Building` can be modeled as a single `MultiSurface`, the remaining surfaces would be `BuildingPart` elements and the aggregated geometry would be a `Solid` (Fig. 7).

Rule 4: A Building or BuildingPart that is modeled as MultiSurface (according to rule 3) must have at least three faces.

```
context Building
inv: self.oclIsTypeOf(t:Building)
inv: self.lod1Solid->isEmpty() and
self.lod1MultiSurface->notEmpty()
inv: self.consistsOfBuildingPart->notEmpty()->
forAll(b:BuildingPart | b.lod1Solid->isEmpty()
and b.lod1Multisurface->notEmpty())
inv: self.lod1MultiSurface->size()≥3
and self.consistsOfBuildingPart.
lod1MultiSurface->size()≥3
```

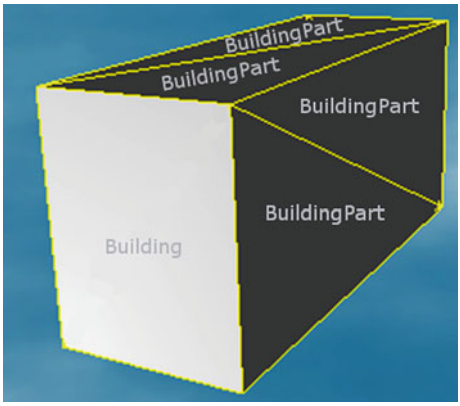


Fig. 7 `Building` and `Buildingparts` modeled as single surfaces

3.7.3 Attributes

Buildings as well as building parts can have further attributes. Some attributes describe geometry-related properties, e.g. `storeysAboveGround`, `storeysHeightsAboveGround` and `measuredHeight`. If the geometry includes underground structures, `storeysBelowGround` and `storeysHeightsBelowGround` can be present as well. In a correct model, their values should be consistent with the geometry. Since these attribute are not unambiguously defined their plausibility should be assessed.

Rule 5: If the number of storeys above ground is given, but not their heights, then a storey height between 2 and 3 meters is assumed. The height of the bounding box of the building geometry is expected to be within the calculated range.

```
context Building
inv: self.storeysHeightsAboveGround.oclIsUndefined()
inv: self.storeysAboveGround*2 <
self.lod1Solid.bboxHeight
< self.storeysAboveGround*3
def bboxHeight:Real
= self.lod1Solid.zCoordinate->maxValue()
- self.lod1Solid.zCoordinate->minValue()
```

The height of the bounding box is decreased by the extent of underground structures included in the model, or in case the `Solid` has been extended towards the ground in order to avoid gaps between the `GroundSurface` and the terrain model. In this case, the part below the lowest point of the `TerrainIntersectionCurve` is subtracted from the height of the bounding box.

```
context Building
inv: self.storeysHeightsAboveGround.oclIsUndefined()
and self.tic->notEmpty()
```

```
inv: self.storeysAboveGround*2 <
self.lod1Solid.zCoordinate- > maxValue()
- self.ticLowestPointHeight <
self.storeysAboveGround*3
def ticLowestPointHeight:Real
= self.tic.zCoordinate->minValue()
```

Rule 6: If `storeysHeightsAboveGround` is given, the height of the bounding box can be calculated more exactly by addition of all values of the list. The number of values should be equal to the number of `storeysAboveGround`.

```
context Building
inv: self.storeysHeightsAboveGround->notEmpty()
inv: self.storeysHeightsAboveGround->size()
= self.storeysAboveGround
inv: self.storeysHeightsAboveGround.sum() =
self.lod1Solid.bboxHeight ± tolerance
def bboxHeight:Real = self.lod1Solid.zCoordinate- >
maxValue() - self.lod1Solid.zCoordinate->minValue()
def: tolerance:Real = 0.5
```

4 Implementation of Test Tool and Validation Library

The presented tests and approaches to detect geometric errors in 3D city models are implemented by a library of check components. This library can be used in order to integrate the check functionality into different software tools and applications. It provides an interface to control which checks are actually performed and allows specifying a certain validation configuration.

The implementation is realized as a standalone Java tool for testing 3D city models for errors (Fig. 8). A Swing-based GUI is put on top of the functionality provided by the check library. Results are written to a log-file as well as displayed in the GUI (Fig. 9).

One example for the integration into an existing software tool is depicted in Fig. 10. Here the check library is integrated into FME (Safe Software 2011) as a custom transformer. The transformer takes *FMEFeatures* as input and checks the features using the check configuration specified by the user. Internally, *FMEFeatures* are converted into a data structure of the library and the checks are performed. As a result the transformer is providing two output channels: one for features without errors (original) and one for features not compliant with the validation rules (errors).

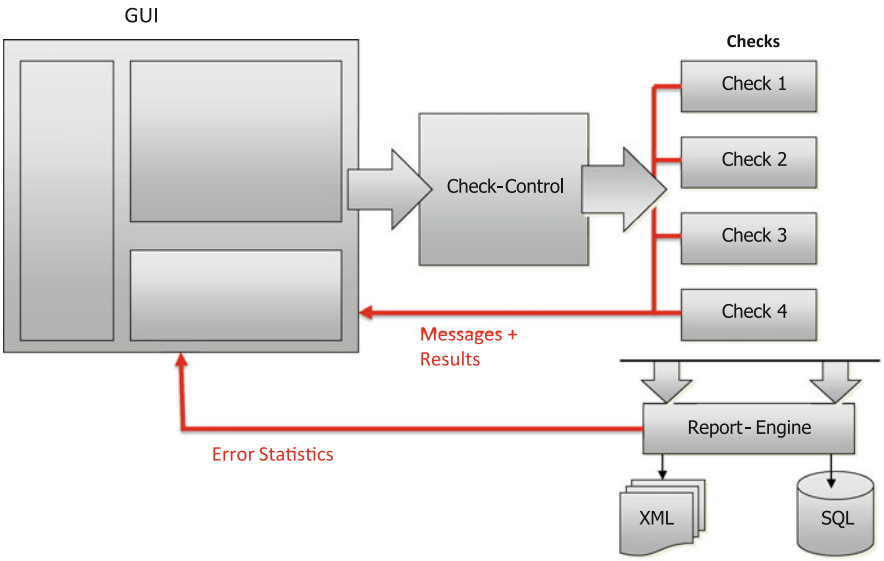


Fig. 8 Stand-alone implementation of the validation tool

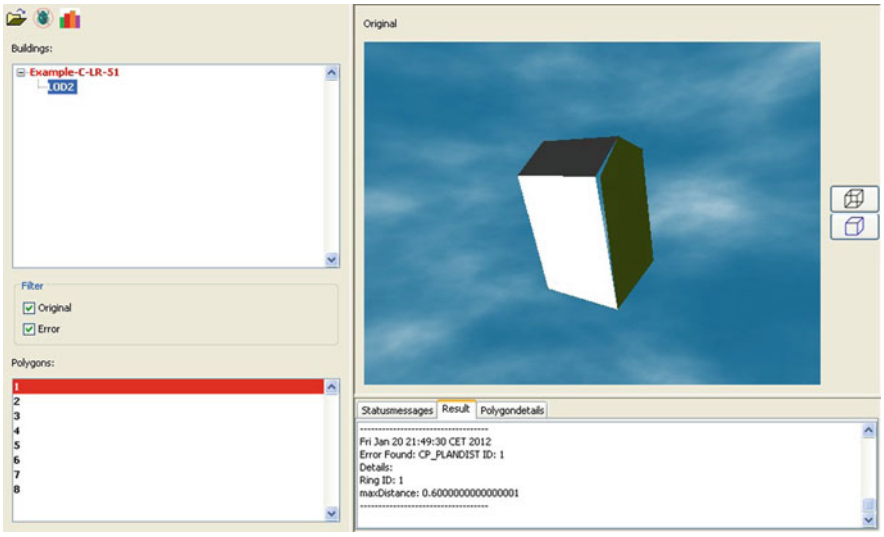


Fig. 9 Graphical user interface (GUI) of the stand-alone validation tool

Other plug-ins for test purposes have been developed for SupportGIS-3D of CPA Geo-Information (CPA-Systems GmbH 2011) and CityServer3D (Coors and Krämer 2011).

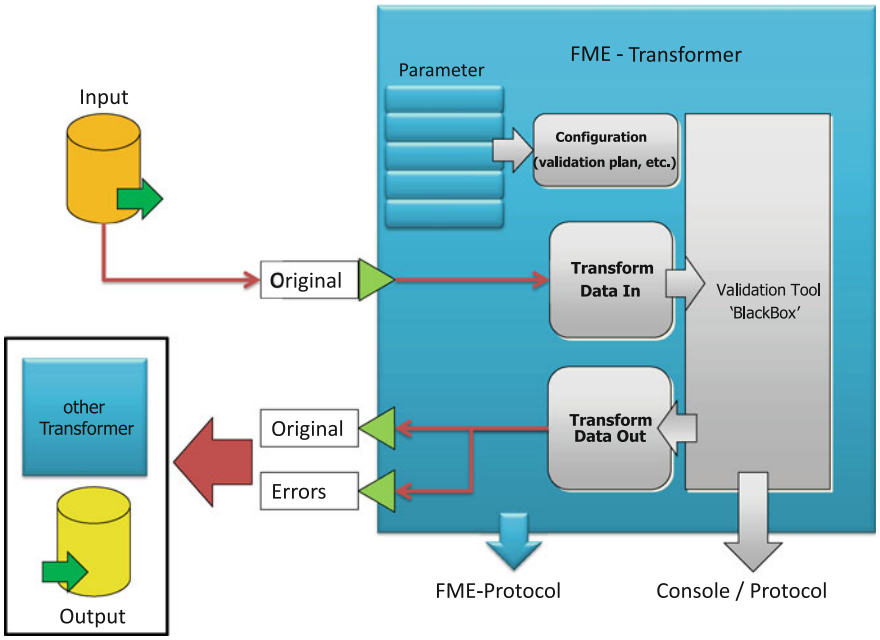


Fig. 10 Integration of the validation tool in FME

Table 3 Characteristics of test models

	A1	A2	H	J	K	L	Q	X
LOD	1	2	1	2	2	2	2	2
# buildings	1	1	61	61	551	4	7	1,922
# polygons	220	580	689	3,455	4251	69	403	32,546
# RoofSurface	–	273	–	1,483	772	10	73	4,957
# WallSurface	–	305	–	1,448	3122	54	25	24,067
# GroundSurface	–	2	–	524	360	5	305	3,522
# undefined	–	0	–	0	0	0	0	0
# edges	654	1402	1701	6,452		182	740	75,370
# vertices	436	779	1134	2,918		123	466	49,690
# holes	0	165	46	1,5504		0	664	0
# polygons per building	220	580	11.3	56.6	7.7	17.3	57.6	16.9

5 Test Results of Real-World Models

The outlined geometry checks of Sect. 3.3 are implemented with JAVA in a standalone application. It is tested against specially created models which contain certain types of errors as well as real-world models or extracts thereof. The models are characterized with certain key figures in Table 3; the validation results are given in Table 4.

Table 4 Error distribution for the test models

Check ID	Axiom (cf. p.7)	A1	A2	H	J	K	L	Q	X
CP-NUMPOINTS	a	0	0	0	0	0	0	0	0
CP-CLOSE	b	0	0	0	0	0	0	0	0
CP-DUPPOINT	c	0	0	0	0	46	0	4	0
CP-SELFINT	d	0	0	0	0	90	0	2	0
CP-PLANDIST	e	0	4	0	0	0	4	8	177
CP-PLANDISTALL	e	0	8	0	0	0	4	8	191
CP-PLANTRI	e	0	67	0	0	45	5	62	269
CS-NUMFACES	g	0	0	0	4,445	133	0	0	0
CS-SELFINT	j	0	0	22	–	647	2	–	4,575
CS-2POLYPEREDGE	h	0	155	46	15,484	6570	0	298	467
CS-FACEORIENT	i	0	–	–	–	–	0	–	–
CS-UMBRELLA	l	0	–	–	–	–	0	–	116
CS-CONCOMP	k	0	–	–	–	–	1	–	980

Only in case CS-2POLYPEREDGE is passed, a building is checked by the dependent checks, viz. the numbers for CS-UMBRELLA and CS-CONCOMP in models L and X refer to buildings which have no CS-2POLYPEREDGE errors.

Besides the different results for the planarity checks (cf. Sect. 3.3) the high numbers of CS-NUMFACES and CS-2POLYPEREDGE in model J are noticeable. A closer look at the details reveals that building installations are modeled as `MultiSurface` elements containing only one single polygon each. A `BuildingInstallation` element is currently treated as a `Solid` geometry which explains the high numbers of errors for CS-2POLYPEREDGE also.

6 Conclusion and Future Work

Rules for validation of geometry in 3D city models were defined and implemented in a JAVA library to enable automated processing of 3D city models. The functionality is available as standalone JAVA application and integrated as plug-in for three commercial systems. Validation tests with real-world models showed that nearly all models have geometric errors although they are visually in order. This fact confirms the importance of geometry validation for 3D city models to enable other applications to work correctly.

Error-free and standard-compliant geometry is a prerequisite for semantic validation. We presented a concept of validation rules for geometric-semantical consistency in LoD 1 of CityGML models. The rules are based on the standard and introduce additional useful restrictions which can help to increase the overall quality. They will be implemented as part of the validation tool in future. Furthermore, the rule set will be extended to LoD2 and above. Other CityGML elements with relation to geometry will be added.

CityGML can be extended by generic attributes (Application Domain Extensions—ADE). Some generic attributes have been defined by organizations such as

the German AdV[3] and many municipalities. Initially, these attributes were intended to reflect additional feature properties which cannot be included in 2D data otherwise. For example, AdV has defined roof-type primitives like gable roof, hip roof, shed roof etc. In LoD 2 and higher, the geometry should reflect assigned roof type properties, thus a validation of certain generic attributes should be considered.

Development of consistency rules according to the concept above will be a major task to bring high-quality city models forward.

Acknowledgments The authors would like to thank the SIG-3D quality working group for fruitful discussion and the German Federal Ministry of Education and Research (BMBF) for funding of the project CityDoctor under provision number 17110B10.

References

Attene M, Falcidieno B (2006) Remesh: an nteractive environment to edit and repair triangle meshes. In: IEEE international conference on shape modeling and applications. p 41

Bogdahn J, Coors V (2010) Towards an automated healingof 3D urban models. In Kolbe TH, König G, Nagel C (eds) In: Proceedings of international conference on 3D geoinformation. International archives of photogrammetry, remote sensing and spatial information science. International conference on 3D geoinformation. Shaker Verlag, Aachen, Germany, pp 13–17

Boguslawski P, Gold C, Ledoux H (2011) Modelling and analysing 3D buildings with a primal/dual data structure. ISPRS J Photogram Rem Sens 66(2):188–197

Borodin P, Novotni M, Klein R (2002) Progressive gap closing for mesh repairing. In: Vince J, Earnshaw R (eds) Advances in modelling, animation and rendering, Springer pp 201–213

Butlin G, Stops C (1996) CAD data repair In: 5th international meshing roundtable. pp 7-12

Coors V, Krämer M (2011) Integrating quality management into a 3D geospatial server. In: UDMS 2011. Urban Data Management Society. Delft, p 8

CPA-Systems GmbH (2011) Available at: http://www.cpa-systems.de/

Elysium (2008) CADdoctor. Available at: http://www.elysiuminc.com/Products/caddoctor.asp. Accessed 20 Jan, 2012

Ghawana T, Zlatanova S (2010) Data consistency checks for building a 3D model: a case study of Technical University, Delft campus. Geospatial World, The Netherlands, p 4

Gröger G, Coors V(2010) Rules for validating GML geometries in CityGML. Available at: http://files.sig3d.de/file/20101215_Regeln_GML_final_DE.pdf. Accessed 18 Jan 2012

Gröger G, Plümer L (2009) How to achieve consistency for 3D city models. GeoInformatica 15(1):137–165

Gröger, G. et al (eds) (2008) OpenGIS city geography markup language (CityGML) encoding standard. Available at: http://portal.opengeospatial.org/files/?artifact_id=28802. Accessed 28 June 2011

International Organization for Standardization (ISO) (2002) ISO 19113 standard: geographic information – quality principles. Available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26018

International Organization for Standardization (ISO) (2003a) ISO 19114 standard: geographic information—quality evaluation procedures. Available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26019

[3] Working Group of the surveying agencies of the federal states.

International Organization for Standardization (ISO) (2003b) ISO/CD 19107, Geographic information—spatial schema

International Organization for Standardization (ISO) (2006) ISO/PAS 26183—SASIG product data quality guidelines for the global automotive industry

International TechneGroup (2011) CADfix. Available at: http://www.transcendata.com/products/cadfix/index.htm

Ju T (2009) Fixing geometric errors on polygonal models: a survey. J Comput Sci Technol 24(1):19–29

Karki, S., Thompson, R. & McDougall, K., 2010. Data validation in a 3D cadastre. In Neutens T, Maeyer P (eds) *Developments in 3D geo-Iinformation sciences*. Lecture notes in geoinformation and cartography, Springer Berlin Heidelberg, pp. 92-122

Kazar BM et al (2008) On valid and invalid three-dimensional geometries. In: Oosterom P et al (eds) Advances in 3D geoinformation systems. Springer, Berlin, pp 19–46 (Available at: http://www.springerlink.com/index/10.1007/978-3-540-72135-2_2. Accessed 22 June 2011)

Ledoux H (2011) Topologically consistent 3D city models obtained by extrusion. Int J Geogr Inf Sci 25(4):557–574

Métral C, Falquet G, Cutting-Decelle AF (2009) Towards semantically enriched 3D citymodels: an ontology-based approach. In: Academic track of geoweb 2009—cityscapes, international archives of photogrammetry, remote sensing and spatial information sciences (ISPRS). Vancouver, Canada

Object Management Group (2011) OMG object constraint language (OCL). Available at: http://www.omg.org/spec/OCL/2.3.1/. Accessed 20 Jan 2012

van Oort P (2005) Spatial data quality: from description to application. Optima Grafische Communicatie, The Netherlands

Oosterom P, Quak W, Tilssen T (2005) About invalid, valid and clean polygons. In Fisher PF (ed) Developments in spatial data handling. 11th international symposium on spatial data handling. Leicester, UK: Springer, Berlin, pp 1–16. Available at: http://dx.doi.org/10.1007/3-540-26772-7_1

Open Geospatial Consortium (OGC) (2011) Data quality DWG. Available at: http://www.opengeospatial.org/projects/groups/dqdwg. Accessed 19 Jan 2012

Pelagatti G et al. (2009) From the conceptual design of spatial constraints to their implementation in real systems. In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems. Seattle, pp 448-451, Nov 4-6

Rocchini C et al (2004) The marching intersections algorithm for merging range images. Vis Comput 20:149–164

Safe Software (2011). FME desktop. Available at: http://www.safe.com/inc/vendors/elqNow/elqRedir.htm?ref=http://downloads.safe.com/fme/brochures/FME_Desktop.pdf

SIG-3D Quality Working Group (2012) Modellierungshandbuch Gebäude. Available at: http://www.sig3d.de/index.php?catid=2&themaid=8777960

Stadler A, Kolbe TH (2007) Spatio-semantic coherence in the integration of 3D city models. In: Proceedings of the 5th inter-national symposium on spatial data quality. Enschede. Available at: http://spirit.bv.tu-berlin.de/igg/htdocs/fileadmin/user_upload/ Stadler/SDQ2007_Stadler_Kolbe.pdf

Thompson R, Oosterom P (2011) Modelling and validation of 3D cadastral objects. In: Zlatanova S et al (eds) Urban and regional data management–UDMS annual 2011. CRC Press, Delft

Transcat PLM (2011) Q-Checker. Available at: http://www.transcat-plm.com/software/transcat-software/q-checker.html. (Accessed 24 Jan 2012)

T-Systems International (2011) VALIDAT. Available at: https://servicenet.t-systems.de/validat. Accessed 24 Jan 2012

Wöhler T, Pries M, Stark R (2009) Effiziente Verfahren zur Aufbereitung von Geometriemodellen für die virtuelle Absicherung. In 3. Symposium Geometrisches Modellieren, Visualisieren und Bildverarbeitung. Stuttgart

Yamakawa S, Shimada K (2009) Removing self intersections of a triangular mesh by edge swapping, edge hammering, and face lifting. In 18th international meshing roundtable