Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <2019-10-17>

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal reference number of this OGC® document: YY-nnnrx

Version: 0.1

Category: OGC® Best Practice

Editor: Charles Heazel

OGC City Geography Markup Language (CityGML) Conceptual Model Best Practice

Document type:	OGC® Best Practice
Document subtype:	
Document stage:	Draft
Document language:	English

Table of Contents

i. Abstract

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.2.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, <tags separated by commas>

iii. Preface

NOTE

Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name Affiliation

Chapter 1. Scope

NOTE Insert Scope text here. Give the subject of the document and the aspects of that scope covered by the document.

Chapter 2. Conformance

This Best Practice defines XXXX.

Requirements for N target types are considered: * AAAA * BBBB

Conformance with this Best Practice shall be checked using all the relevant tests specified in Annex A (normative) of this document.

In order to conform to this OGC® Best Practice, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative). * Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the document(s) identified.

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

NOTE

Insert References here. If there are no references, state “There are no normative references”.

References are to follow the Springer LNCS style, with the exception that optional information may be appended to references: DOIs are added after the date and web resource references may include an access date at the end of the reference in parentheses. See examples from Springer and OGC below.

Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)

May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)

Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)

Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)

Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)

National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov

ISO / TC 211: ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals (2014)

ISO / TC 211: ISO 19157:2013 Geographic information — Data quality (2013)

ISO / TC 211: ISO 19139:2007 Geographic information — Metadata — XML schema implementation (2007)

ISO / TC 211: ISO 19115-3: Geographic information — Metadata — Part 3: XML schemas (2016)

OGC: OGC 15-097 OGC Geospatial User Feedback Standard. Conceptual Model (2016)

OGC: OGC 12-019, OGC City Geography Markup Language (CityGML) Encoding Standard (2012)

OGC: OGC 14-005r3, OGC IndoorGML (2014)

Chapter 4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Best Practice.

For the purposes of this document, the following additional terms and definitions apply.

4.1. term name

text of the definition

Chapter 5. Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this document are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

Chapter 6. CityGML Conceptual Model

This section provides a detailed discussion of the CityGML Conceptual Model.

Core

The CityGML Core module defines the basic concepts and components of the overall CityGML data model. It forms the universal lower bound of the CityGML data model and, thus, is a dependency of all extension modules. Consequently, the core module has to be implemented by any conformant system. Primarily, the core module provides the abstract base classes from which thematic classes within extension modules are (transitively) derived. Besides abstract type definitions, the core also contains non-abstract content, for example basic data types and thematic classes that may be used by more than one extension module. The UML diagram in Fig. 21 illustrates CityGML's core module, for the XML Schema definition see below and annex A.1.

NOTE insert Fig 21 UML

The base class of all thematic classes within CityGML's data model is the abstract class _CityObject. _CityObject provides a creation and a termination date for the management of histories of features as well as the possibility to model external references to the same object in other data sets. Furthermore, two qualitative attributes relativeToTerrain and relativeToWater are provided which enable to specify the feature's location with respect to the terrain and water surface. The possible topological relations are illustrated in Fig. 22. Both attributes facilitate simple and efficient queries like for the number of subsurface buildings (entirelyBelowTerrain) without the need for an additional digital terrain model or a model of the water body.

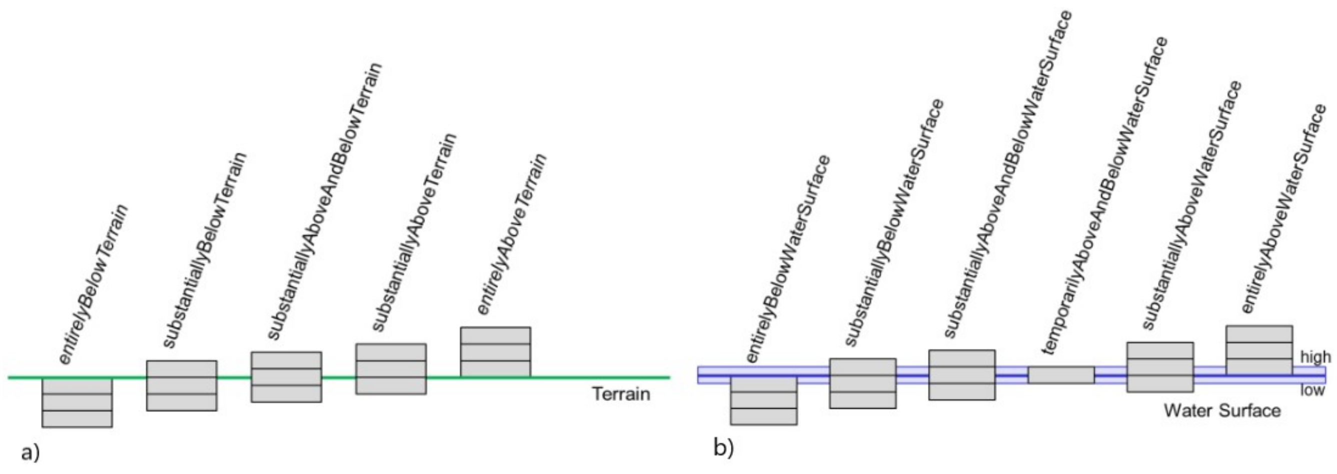
entirelyBelowTerrain
substantiallyBelowTerrain
substantiallyAboveAndBelowTerrain
substantiallyAboveTerrain
entirelyAboveTerrain
Terrain
a)
entirelyBelowWaterSurface
substantiallyBelowWaterSurface
substantiallyAboveAndBelowWaterSurface
temporarilyAboveAndBelowWaterSurface
substantiallyAboveWaterSurface
entirelyAboveWaterSurface
high
low
Water Surface
b)

Figure 1. Topological relations of a CityGML object with respect to a) the terrain and b) the water surface.

_CityObject is a subclass of the GML class _Feature, thus it inherits the metadata property (which can be e.g. information about the lineage, quality aspects, accuracy, local CRS) and name property from the superclass _GML. A _CityObject may have multiple names, which are optionally qualified by a codeSpace. This enables the differentiation between, for example, an official name and a popular name or of names in different languages (cf. the name property of GML objects, Cox et al. 2004). The generalisation property generalizesTo of _CityObject may be used to relate features, which represent the same real-world object in different Levels-of-Detail, i.e. a feature and its generalised counterpart(s). The direction of this relation is from the feature to the corresponding generalised feature.

Thematic classes may have further subclasses with relations, attributes and geometry. Features of the specialized subclasses of _CityObject may be aggregated to a single CityModel, which is a feature collection with optional metadata. Generally, each feature has the attributes class, function, and usage, unless it is stated otherwise. The class attribute can occur only once, while the attributes usage and function can be used multiple times. The class attribute allows for the classification of features beyond the thematic class hierarchy of _CityObject. For exam-ple, a building feature is represented by the thematic subclass bldg:Building of _CityObject in the first place (this subclass is defined within CityGML's Building module, cf. chapter 10.3). A further classification, e.g. as resi-dential or administration building, may then be modelled using the class attribute of the class bldg:Building. The attribute function normally denotes the intended purpose or usage of the object, such as hotel or shopping centre for a building, while the attribute usage normally defines its real or actual usage. Possible values for the attrib-utes class, function, and usage can be specified in code lists which are recommended to be implemented as simple dictionaries following the Simple Dictionary Profile of GML 3.1.1 (cf. chapter 6.6 and 10.14). Annex C provides code lists proposed and maintained by the SIG 3D which contain feasible attribute values and which may be extended or redefined by users.

In addition to thematic content, the core module also provides the concept of implicit geometries as an enhance-ment of the geometry model of GML3. Since this concept is strongly related to the spatial model of CityGML it has already been introduced in chapter 8.2.

The top level class hierarchy of the thematic model in CityGML is presented in Fig. 23. The subclasses of _CityObject comprise the different thematic fields of a city model covered by separate CityGML extension modules: the terrain, buildings, bridges, tunnels, the coverage by land use objects, water bodies, vegetation, generic city objects, city furniture objects, city object groups, and transportation. To indicate the extension module defining a respective subclass of _CityObject, the class names in Fig. 23 are preceded by prefixes. Each prefix is associated with one CityGML extension module (see chapter 4.3 and chapter 7 for a list of CityGML's extension modules and the corresponding prefixes).

NOTE Insert Fig 23 UML

The classes GenericCityObject and _genericAttribute defined within CityGML's Generics module (cf. chapters 6.11 and 10.12) allow for modelling and exchanging of 3D objects which are not covered by any other thematic class or which require attributes not represented in CityGML. For example, in the future, sites derived from the abstract class _Site of the core module may be completed by further subclasses like excavation, city wall or embankment. At present, the class GenericCityObject should be used in order to represent and exchange these features. However, the concept of generic city objects and attributes may only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

If the Generics module is employed, each CityObject may be assigned an arbitrary number of generic attributes in order to represent additional properties of features. For this purpose, the

Generics module augments the abstract base class _CityObject by the property element _genericAttribute. The additional property _genericAttribute is injected into _CityObject using CityGML's Application Domain Extension mechanism (cf. chapter 10.13). By this means, each thematic subclass of _CityObject inherits this property and, thus, the possi-bility to contain generic attributes. Therefore, the Generics module has a deliberate impact on all CityGML extension modules defining thematic subclasses of _CityObject.

Appearance information about a feature's surfaces can be represented by the class Appearance provided by CityGML's Appearance module (cf. chapter 9). In contrast to the other thematic extensions to the core, Appear-ance is not derived from _CityObject but from the GML class _Feature. _CityObject features and Appearance features may be embraced within a single CityModel feature collection in an arbitrary or even mixed sequence using the cityObjectMember and appearanceMember elements, both being members of the substitution group gml:featureMember (cf. chapter 9 and chapter 10.1.1). Furthermore, feature appearances may be stored inline the _CityObject itself. In order to enable city objects to store appearance information, the Appearance module augments the abstract base class _CityObject by the property element appearance using CityGML's Application Domain Extension mechanism (cf. chapter 10.13). Consequently, the appearance property is only available for _CityObject and its thematic subclasses if the Appearance module is supported. Therefore, like the Generics module, the Appearance module has a deliberate impact on any other extension module.

For sake of completeness, the class TexturedSurface is also illustrated in Fig. 23. This approach of appearance modelling of previous versions of CityGML has been deprecated and is expected to be removed in future CityGML versions. Since the information covered by TexturedSurface can be losslessly converted to the Ap-pearance module, the use of TexturedSurface is strongly discouraged.

Base elements

AbstractCityObjectType, _CityObject

AbstractCityObjectType, _CityObject

NOTE insert AbstractCityObjectType, _CityObject UML

CityModelType, CityModel

NOTE insert CityModelType, CityModel UML

cityObjectMember

NOTE insert cityObjectMember UML

AbstractSiteType, _Site

NOTE insert AbstractSiteType, _Site UML

The abstract class _Site is intended to be the superclass for buildings, bridges, tunnels, facilities, etc. Future extension of CityGML (e.g. excavations, city walls or embankments) would be modelled as subclasses of _Site. As subclass of _CityObject, a _Site inherits all attributes and relations, in particular the id, names, external references, and generalisation relations.

Generalisation relation, RelativeToTerrainType and RelativeToWaterType

GeneralizationRelationType

NOTE insert GeneralizationRelationType UML

RelativeToTerrainType, RelativeToWaterType

NOTE insert RelativeToTerrainType, RelativeToWaterType UML

External references

An ExternalReference defines a hyperlink from a _CityObject to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS MasterMap®. The reference consists of the name of the external information system, represented by an URI, and the reference of the external object, given either by a string or by an URI. If the informationSystem element is missing in the ExternalReference, the ExternalObjectReference must be an URI.

ExternalReferenceType, ExternalObjectReferenceType

NOTE insert ExternalReferenceType, ExternalObjectReferenceType UML

Address information

The CityGML core module provides the means to represent address information of real-world features within virtual city models. Since not every real-world feature is assigned an address, a correspondent address property is not defined for the base class _CityObject, but has to be explicitly modelled for a thematic subclass. For example, the building model declares address properties for its classes _AbstractBuilding and Door. Both classes are referencing the corresponding data types of the core module to represent address information (cf. chapter 10.3).

Addresses are modelled as GML features having one xalAddress property and an optional multiPoint property. For example, for a building feature the multiPoint property allows for the specification of the exact positions of the building entrances that are associated with the corresponding address. The point coordinates can be 2D or 3D. Modelling addresses as features has the advantage that GML3’s method of representing features by refer-ence (using XLinks) can be applied. This means, that addresses might be bundled as an address FeatureCollec-tion that is stored within an external file or that can be served by an external Web Feature Service. The address property elements within the CityGML file then would not contain the address information inline but only references to the corresponding external features.

The address information is specified using the xAL address standard issued by the OASIS consortium (OASIS 2003), which provides a generic schema for all kinds of international addresses. Therefore, child elements of the xalAddress property of Address have to be structured according to the OASIS xAL schema.

AddressPropertyType, AddressType, Address

NOTE insert AddressPropertyType, AddressType, Address UML

The following two excerpts of a CityGML dataset contain examples for the representation of German and British addresses in xAL. The address information is attached to building objects (bldg:Building) according to the CityGML Building module (cf. chapter 10.3). Generally, if a CityGML instance document contains address information, the namespace prefix “xAL” should be declared in the root element and must refer to “urn:oasis:names:tc:ciq:xsdschema:xAL:2.0”. An example showing a complete CityGML dataset including a building with an address element is provided in annex G.1.

NOTE insert examples here if appropriate.

Digital Terrain Model

An essential part of a city model is the terrain. The Digital Terrain Model (DTM) of CityGML is provided by the thematic extension module Relief (cf. chapter 7). In CityGML, the terrain is represented by the class ReliefFea-ture in LOD 0-4 (Fig. 24 depicts the UML diagram, for the XML schema definition see annex A.9). A Re-liefFeature consists of one or more entities of the class ReliefComponent. Its validity may be restricted to a certain area defined by an optional validity extent polygon. As ReliefFeature and ReliefComponent are deriva-tives of _CityObject, the corresponding attributes and relations are inherited. The class ReliefFeature is associat-ed with different concepts of terrain representations which can coexist. The terrain may be specified as a regular raster or grid (RasterRelief), as a TIN (Triangulated Irregular Network, TINReflief), by break lines (BreaklineRe-lief), or by mass points (MasspointRelief). The four types are implemented by the corresponding GML3 classes: grids by gml:RectifiedGridCoverage, break lines by gml:MultiCurve, mass points by gml:MultiPoint and TINs either by gml:TriangulatedSurface or by gml:Tin. In case of gml:TriangulatedSurfaces, the triangles are given explicitly while in case of gml:Tin only 3D points are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation, cf. Okabe et al. 1992). Break lines are represented by 3D curves. Mass points are simply a set of 3D points.

NOTE insert Fig 24 UML

In a CityGML dataset the four terrain types may be combined in different ways, yielding a high flexibility. First, each type may be represented in different levels of detail, reflecting different accuracies or resolutions. Second, a part of the terrain can be described by the combination of multiple types, for example by a raster and break lines, or by a TIN and break lines. In this case, the break lines must share the geometry with the triangles. Third, neighboring regions may be represented by different types of terrain models. To facilitate this combination, each terrain object is provided with a spatial attribute denoting its extent of validity (Fig. 25). In most cases, the extent of validity of a regular raster dataset corresponds to its bounding box. This validity extent is represented by a 2D footprint polygon, which may have holes. This concept enables, for example, the modelling of a terrain by a coarse grid, where some distinguished regions are represented by a detailed, high-accuracy TIN. The boundaries between both types are given by the extent attributes of the corresponding terrain objects.

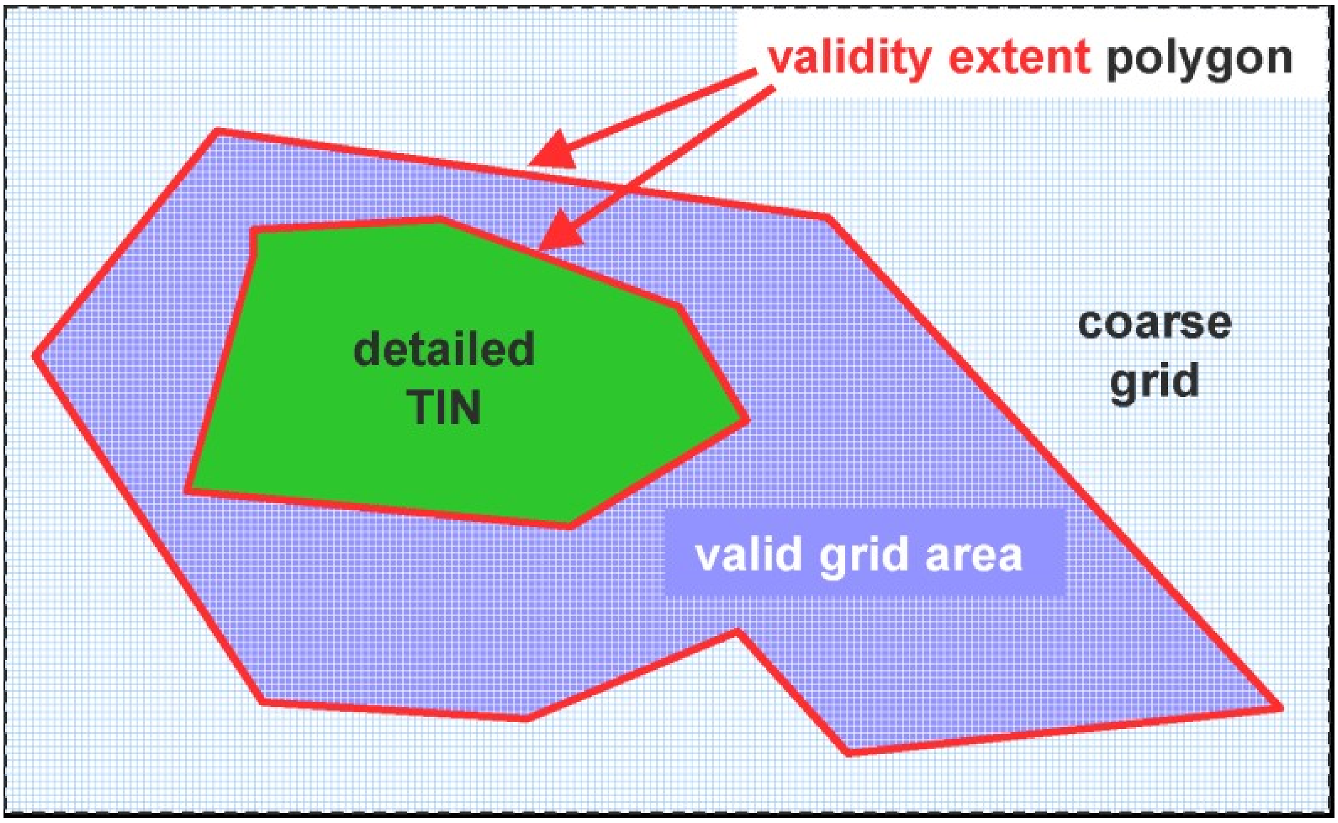

Figure 2. Nested DTMs in CityGML using validity extent polygons (graphic: IGG Uni Bonn).

Accuracy and resolution of the DTM are not necessarily dependent on features of other CityGML extenstion modules such as building models. Hence, there is the possibility to integrate building models with higher LOD to a DTM with lower accuracy or resolution.

This approach interacts with the concept of TerrainIntersectionCurves TIC (cf. chapter 6.5). The TIC can be used like break lines to adjust the DTM to different features such as buildings, bridges, or city furnitures, and hence to ensure a consistent representation of the DTM. If necessary, a retriangulation may have to be processed. A TIC can also be derived by the individual intersection of the DTM and the corresponding feature.

ReliefFeature and its ReliefComponents both have an lod attribute denoting the corresponding level of detail. In most cases, the LOD of a ReliefFeature matches the LOD of its ReliefComponents. However, it is also allowed to specify a ReliefFeature with a high LOD which consists of ReliefComponents where some of them can have a LOD lower than that of the aggregating ReliefFeature. The idea is that, for example, for a LOD3 scene it might be sufficient to use a regular grid in LOD2 with certain higher precision areas defined by ReliefComponents in LOD3. The LOD2 grid and the LOD3 components can easily be integrated using the concept of the validity extent polygon. Therefore, although some of the ReliefComponents would have been classified to a lower LOD, the whole ReliefFeature would be appropriate to use with other LOD3 models which is indicated by setting its lod value to 3.

Relief feature and relief component

ReliefFeatureType, ReliefFeature

NOTE insert ReliefFeatureType, ReliefFeature UML

AbstractReliefComponentType, _ReliefComponent

NOTE insert AbstractReliefComponentType, _ReliefComponent UML

TIN relief

TINReliefType, TINRelief

NOTE insert TINReliefType, TINRelief UML

The geometry of a TINRelief is defined by the GML geometry class gml:TriangulatedSurface. This allows either the explicit provision of a set of triangles (gml:TriangulatedSurface) or specifying of only the control points, break and stop lines using the subclass gml:Tin of gml:TriangulatedSurface. In the latter case, an application that processes an instance document containing a gml:Tin has to reconstruct the triangulated surface by the applica-tion of a constrained Delaunay triangulation algorithm (cf. Okabe et al. 1992).

Raster relief

RasterReliefType, RasterRelief, Elevation

NOTE insert RasterReliefType, RasterRelief, Elevation UML

Mass point relief

MassPointReliefType, MassPointRelief

NOTE insert MassPointReliefType, MassPointRelief UML

Breakline relief

BreaklineReliefType, BreaklineRelief

NOTE insert BreaklineReliefType, BreaklineRelief UML

The geometry of a BreaklineRelief can be composed of break lines and ridge/valley lines. Whereas break lines indicate abrupt changes of terrain slope, ridge/valley lines in addition mark a change of the sign of the terrain slope gradient. A BreaklineRelief must have at least one of the two properties.

Building Model

The building model is one of the most detailed thematic concepts of CityGML. It allows for the representation of thematic and spatial aspects of buildings and building parts in five levels of detail, LOD0 to LOD4. The building model of CityGML is defined by the thematic extension module Building (cf. chapter 7). Fig. 26 provides examples of 3D city and building models in LOD1 – 4.

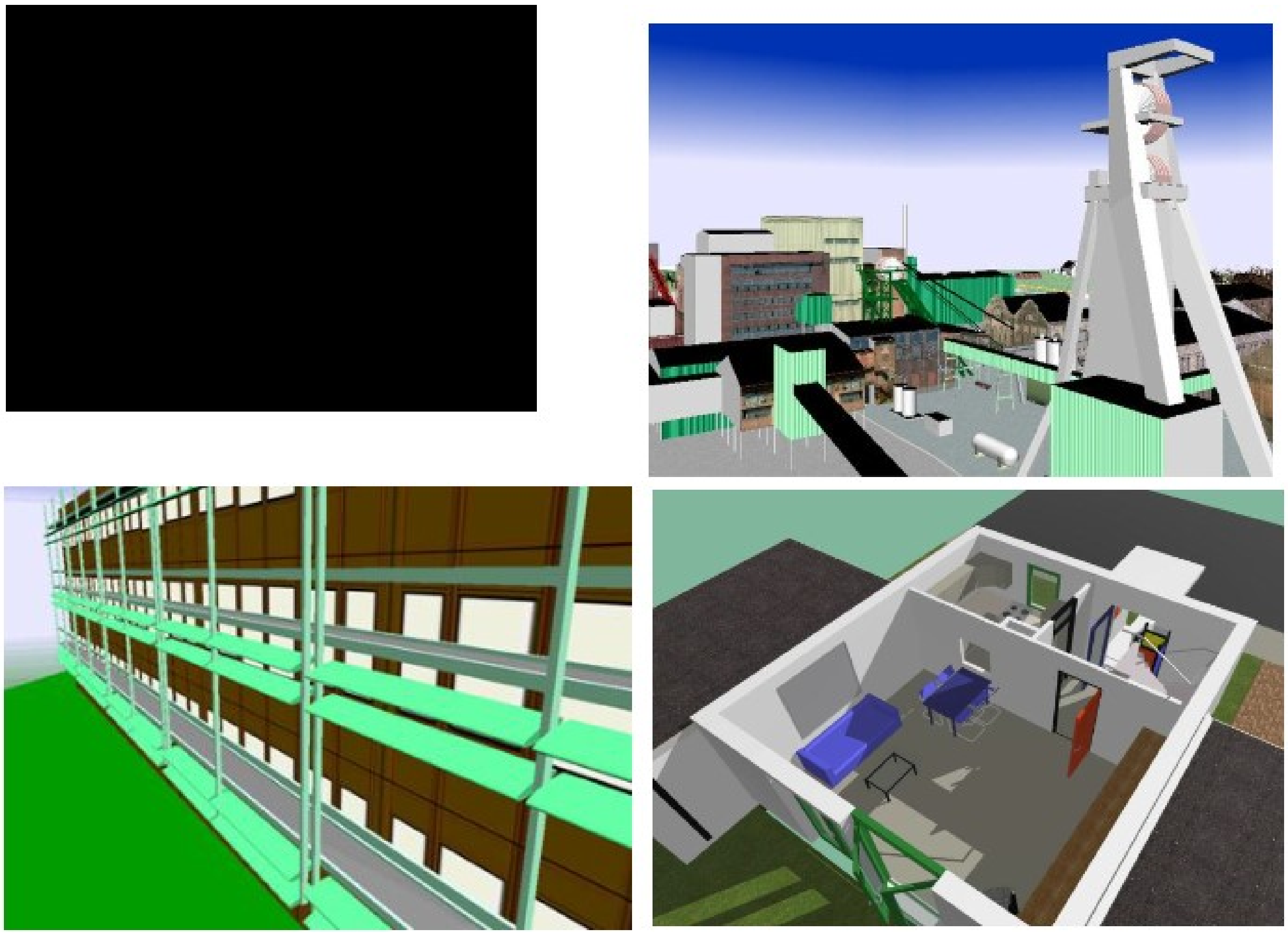

Figure 3. Examples for city or building models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left), and LOD4 (lower right) (source: District of Recklinghausen, m-g-h ingenieure+architekten GmbH).

The UML diagram of the building model is depicted in Fig. 27, for the XML schema definition see annex A.4 and below. The pivotal class of the model is _AbstractBuilding, which is a subclass of the thematic class _Site (and transitively of the root class _CityObject). _AbstractBuilding is specialised either to a Building or to a BuildingPart. Since an _AbstractBuilding consists of BuildingParts, which again are _AbstractBuildings, an aggregation hierarchy of arbitrary depth may be realised. As subclass of the root class _CityObject, an _AbstractBuilding inherits all properties from _CityObject like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences (cf. chapter 6.7). Further properties not explicitly covered by _AbstractBuilding may be modelled as generic attributes provided by the CityGML Generics module (cf. chapter 10.12) or using the CityGML Application Domain Extension mechanism (cf. chapter 10.13).

Building complexes, which consist of a number of distinct buildings like a factory site or hospital complex, should be aggregated using the concept of CityObjectGroups (cf. chapter 6.8). The main building of the complex can be denoted by providing “main building” as the role name of the corresponding group member.

Both classes Building and BuildingPart inherit the attributes of _AbstractBuilding: the class of the building, the function (e.g. residential, public, or industry), the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individual heights of the storeys above and below ground. This set of parameters is suited for roughly reconstructing the three-dimensional shape of a building and can be provided by cadastral systems. Furthermore, Address features can be assigned to Buildings or BuildingParts.

NOTE insert Fig 27 UML

The geometric representation and semantic structure of an _AbstractBuilding is shown in Fig. 27. The model is successively refined from LOD0 to LOD4. Therefore, not all components of a building model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the proposed minimum acquisition criteria for each LOD (cf. chapter 6.2). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

In LOD0, the building can be represented by horizontal, 3-dimensional surfaces. These can represent the foot-print of the building and, separately, the roof edge. This allows the easy integration of 2D data into the model. In many countries these 2D geometries readily exist, for example in cadastral or topographic data holdings. Cadas-tre data typically depicts the shape of the building on the ground (footprints) and topographic data is often a mixture between footprints and geometries at roof level (roof edges), which are often photogrametrically extract-ed from area/satellite images or derived from airborne laser data. The building model allows the inclusion of both. In this case large overhanging roofs can be modelled as a preliminary stage to more detailed LOD2 and LOD3 depictions. The surface geometries require 3D coordinates, though it is mandated that the height values of all vertices belonging to the same surface are identical. If 2D geometries are imported into any of these two LOD0 geometries, an appropriate height value for all vertices needs to be chosen. The footprint is typically located at the lowest elevation of the ground surface of the building whereas the roof edge representation should be placed at roof level (e.g., eaves height).

In LOD1, a building model consists of a generalized geometric representation of the outer shell. Optionally, a gml:MultiCurve representing the TerrainIntersectionCurve (cf. chapter 6.5) can be specified. This geometric representation is refined in LOD2 by additional gml:MultiSurface and gml:MultiCurve geometries, used for modelling architectural details like roof overhangs, columns, or antennas. In LOD2 and higher LODs the outer facade of a building can also be differentiated semantically by the classes _BoundarySurface and BuildingInstal-lation. A _BoundarySurface is a part of the building’s exterior shell with a special function like wall (WallSur-face), roof (RoofSurface), ground plate (GroundSurface), outer floor (OuterFloorSurface), outer ceiling (Outer-CeilingSurface) or ClosureSurface. The BuildingInstallation class is used for building elements like balconies, chimneys, dormers or outer stairs, strongly affecting the outer appearance of a building. A BuildingInstallation may have the attributes class, function, and usage (cf. Fig. 27).

In LOD3, the openings in _BoundarySurface objects (doors and windows) can be represented as thematic ob-jects. In LOD4, the highest level of resolution, also the interior of a building, composed of several rooms, is represented in the building model by the class Room. This enlargement allows a virtual accessibility of buildings, e.g. for visitor information in a museum (“Location Based Services“), the examination of accommodation standards or the presentation of daylight illumination of a building. The aggregation of rooms according to arbitrary, user defined criteria (e.g. for defining the rooms corresponding to a certain storey) is achieved by employing the general grouping concept provided by CityGML (cf. chapter 10.3.6). Interior installations of a building, i.e.

objects within a building which (in contrast to furniture) cannot be moved, are represented by the class IntBuildingInstallation. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the Room class, otherwise (e.g. in case of rafters or pipes) with _AbstractBuilding. A Room may have the attributes class, function and usage whose value can be defined in code lists (chapter 10.3.8 and annex C.1). The class attribute allows a classification of rooms with respect to the stated function, e.g. commercial or private rooms, and occurs only once. The function attribute is intended to express the main purpose of the room, e.g. living room, kitchen. The attribute usage can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The visible surface of a room is represented geometrically as a Solid or MultiSurface. Semantically, the surface can be structured into specialised _BoundarySurfaces, representing floor (FloorSurface), ceiling (CeilingSur-face), and interior walls (InteriorWallSurface). Room furniture, like tables and chairs, can be represented in the CityGML building model with the class BuildingFurniture. A BuildingFurniture may have the attributes class, function and usage. Annexes G.1 to G.6 provide example CityGML documents containing a single building model which is subsequently refined from a coarse LOD0 representation up to a semantically rich and geomet-ric-topologically sound LOD4 model including the building interior.

Building and Building Part

NOTE Version 2.0 uses XML schema to illustrate this section. Replace those schema with UML.

BuildingType, Building

NOTE Insert BuildingType, Building UML

The Building class is one of the two subclasses of _AbstractBuilding. If a building only consists of one (homo-geneous) part, this class shall be used. A building composed of structural segments differing in, for example the number of storeys or the roof type has to be separated into one Building having one or more additional Build-ingPart (see Fig. 28). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating Building feature.

BuildingType, Building Part

NOTE Insert BuildingType, Building Part UML

The class BuildingPart is derived from _AbstractBuilding. It is used to model a structural part of a building (see Fig. 28). A BuildingPart object should be uniquely related to exactly one building or building part object.

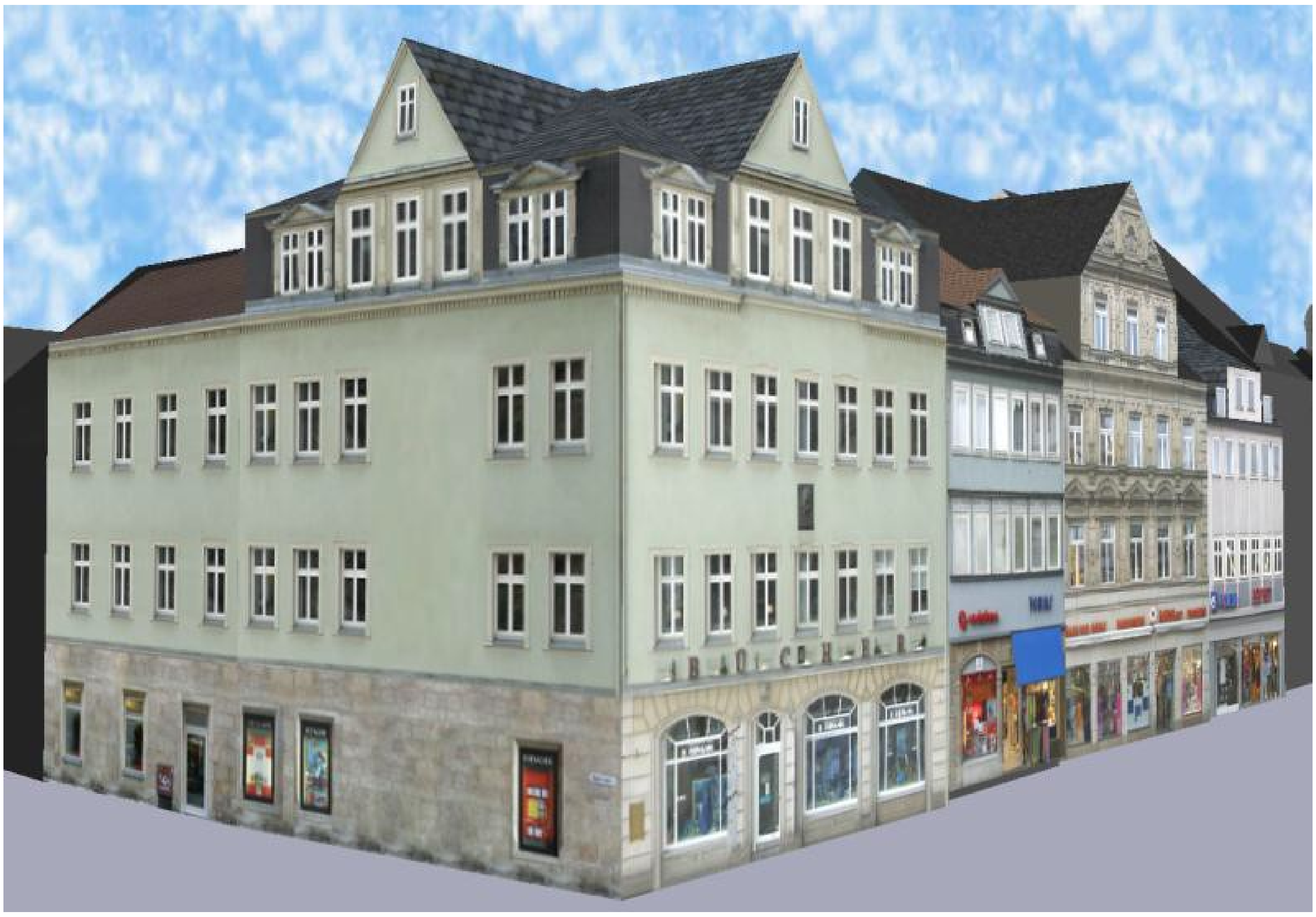

Figure 4. Examples of buildings consisting of one and two building parts (source: City of Coburg)

AbstractBuildingType, _AbstractBuilding

NOTE AbstractBuildingType, _AbstractBuilding UML

The abstract class _AbstractBuilding contains properties for building attributes, purely geometric representations, and geometric/semantic representations of the building or building part in different levels of detail. The attributes describe:

1. classification of the building or building part (class), the different intended usages (function), and the different actual usages (usage). The permitted values for these attributes can be specified in code lists.
2. The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the build-ing or building part. These attributes can be used to describe the chronology of the building development within a city model. The points of time refer to real world time.
3. The roof type of the building or building part (roofType). The permitted values for this attribute can be specified in a code list.
4. The measured relative height (measuredHeight) of the building or building part.
5. The number of storeys above (storeyAboveGround) and below (storeyBelowGround) ground level.
6. The list of storey heights above (storeyHeightsAboveGround) and below (storeyHeightsBelowGround) ground level. The first value in a list denotes the height of the nearest storey wrt. to the ground level and last value the height of the farthest.

Spanning the different levels of detail, the building model differs in the complexity and granularity of the geo-metric representation and the thematic structuring of the model into components with a special semantic mean-ing. This is illustrated in Fig. 29 and Fig. 30, showing the same building in five different LODs. The class _AbstractBuilding has a number of properties which are associated with certain LODs.

NOTE insert Fig 29

NOTE inset Fig 30

Tab. 5 shows the correspondence of the different geometric and semantic themes of the building model to LODs. In LOD1 – 4, the volume of a building can be expressed by a gml:Solid geometry and/or a gml:MultiSurface geometry. The definition of a 3D Terrain Intersection Curve (TIC), used to integrate buildings from different sources with the Digital Terrain Model, is also possible in LOD1 – 4. The TIC can – but does not have to – build closed rings around the building or building parts.

In LOD0 (cf. Fig. 29) the building is represented by horizontal surfaces describing the footprint and the roof edge.

In LOD1 (cf. Fig. 30), the different structural entities of a building are aggregated to a simple block and not differentiated in detail. The volumetric and surface parts of the exterior building shell are identical and only one of the corresponding properties (lod1Solid or lod1MultiSurface) must be used.

In LOD2 and higher levels of detail, the exterior shell of a building is not only represented geometrically as gml:Solid geometry and/or a gml:MultiSurface geometry, but it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is _BoundarySurface (cf. chapter 10.3.2), which is associated with a gml:MultiSurface geometry. If in a building model there is both a geometric represen-tation of the exterior shell as volume or surface model and a semantic representation by means of thematic _BoundarySurfaces, the geometric representation must not explicitly define the geometry, but has to reference the corresponding geometry components of the gml:MultiSurface of the _BoundarySurface elements.

Table 1. Semantic themes of the class _AbstractBuilding

Geometric / semantic theme	Property type	LOD0	LOD1	LOD2	LOD3	LOD4
Building footprint and roof edge	gml:MultiSurfaceType	•				
Volume part of the building shell	gml:SolidType		•	•	•	•
Surface part of the building shell	gml:MultiSurfaceType		•	•	•	•
Terrain intersection curve	gml:MultiCurveType		•	•	•	•
Curve part of the building shell	gml:MultiCurveType			•	•	•

Geometric / semantic theme	Property type	LOD0	LOD1	LOD2	LOD3	LOD4
Building parts	BuildingPartType		•	•	•	•
Boundary surfaces (chapter 10.3.3)	AbstractBoundary SurfaceType			•	•	•
Outer building installations (chapter 10.3.2)	BuildingInstallationType			•	•	•
Openings (chapter 10.3.4)	AbstractOpeningType				•	•
Rooms (chapter 10.3.5)	RoomType					•
Interior building installations (chapter 10.3.5)	IntBuildingInstallationType					•

Apart from BuildingParts, smaller features of the building ("outer building installations") can also strongly affect the building characteristic. These features are modelled by the class BuildingInstallation (cf. chapter 10.3.2). Typical candidates for this class are chimneys (see. Fig. 30), dormers (see Fig. 28), balconies, outer stairs, or antennas. BuildingInstallations may only be included in LOD2 models, if their extents exceed the proposed minimum dimensions as specified in chapter 6.2. For the geometrical representation of the class Build-ingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used.

The class _AbstractBuilding has no additional properties for LOD3. Besides the higher requirements on geomet-ric precision and smaller minimum dimensions, the main difference of LOD2 and LOD3 buildings concerns the class _BoundarySurface (cf. chapter 10.3.3). In LOD3, openings in a building corresponding with windows or doors (see Fig. 30) are modelled by the abstract class _Opening and the derived subclasses Window and Door (cf. chapter 10.3.4).

With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. But LOD4 pro-vides the possibility to model the interior structure of a building with the classes IntBuildingInstallation and Room (cf. chapter 10.3.5).

Each Building or BuildingPart feature may be assigned zero or more addresses using the address property. The corresponding AddressPropertyType is defined within the CityGML core module (cf. chapter 10.1.4).

Outer building installations

BuildingInstallationType, BuildingInstallation

Note: insert BuildingInstallation UML

A BuildingInstallation is an outer component of a building which has not the significance of a BuildingPart, but which strongly affects the outer characteristic of the building. Examples are chimneys, stairs, antennas, balconies or attached roofs above stairs and paths. A

BuildingInstallation optionally has attributes class, function and usage. The attribute class - which can only occur once - represents a general classification of the installation. With the attributes function and usage, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of a BuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alterna-tively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype building installation is stored only once in a local coordinate system and referenced by other building installation features (see chapter 8.2). The visible surfaces of a building installation can be seman-tically classified using the concept of boundary surfaces (cf. 10.3.3). A BuildingInstallation object should be uniquely related to exactly one building or building part object.

6.1. Boundary surfaces

6.1.1. AbstractBoundarySurfaceType, _BoundarySurface

NOTE Insert AbstractBoundarySurfaceType, _BoundarySurface UML

_BoundarySurface is the abstract base class for several thematic classes, structuring the exterior shell of a build-ing as well as the visible surfaces of rooms and both outer and interior building installations. It is a subclass of _CityObject and thus inherits all properties like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences. From _BoundarySurface, the thematic classes RoofSur-face, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface, ClosureSurface, FloorSurface, InteriorWallSurface, and CeilingSurface are derived. The thematic classification of building surfaces is illustrat-ed in Fig. 31 (outer building shell) and Fig. 32 (additional interior surfaces) and subsequently specified.

For each LOD between 2 and 4, the geometry of a _BoundarySurface may be defined by a different gml:MultiSurface geometry.

In LOD3 and LOD4, a _BoundarySurface may contain _Openings (cf. chapter 10.3.4) like doors and windows. If the geometric location of _Openings topologically lies within a surface component (e.g. gml:Polygon) of the gml:MultiSurface geometry, these _Openings must be represented as holes within that surface. A hole is repre-sented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface’s normal vector). If such an opening is sealed by a Door, a Window, or a ClosureSurface, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an Opening belong to the relevant adjacent _BoundarySurface. If, for example a door seals the Opening, the embrasure surface on the one side of the door belongs to the InteriorWallSurface and on the other side to the WallSurface (Fig. 32 on the right).

NOTE insert Fig 31

NOTE insert Fig 32

6.1.2. GroundSurfaceType, GroundSurface

NOTE insert GroundSurfaceType, GroundSurface uml

The ground plate of a building or building part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the building’s footprint. However, the surface normal of the ground plate is pointing downwards.

6.1.3. OuterCeilingSurfaceType, OuterCeilingSurface

NOTE insert OuterCeilingSurfaceType, OuterCeilingSurface UML

A mostly horizontal surface belonging to the outer building shell and having the orientation pointing downwards can be modeled as an OuterCeilingSurface. Examples are the visible part of the ceiling of a loggia or the ceiling of a passage.

6.1.4. WallSurfaceType, WallSurface

NOTE insert WallSurfaceType, WallSurface UML

All parts of the building facade belonging to the outer building shell can be modelled by the class WallSurface.

6.1.5. OuterFloorSurfaceType, OuterFloorSurface

NOTE insert OuterFloorSurfaceType, OuterFloorSurface UML

A mostly horizontal surface belonging to the outer building shell and with the orientation pointing upwards can be modeled as an OuterFloorSurface. An example is the floor of a loggia.

6.1.6. RoofSurfaceType, RoofSurface

NOTE insert RoofSurfaceType, RoofSurface UML

The major roof parts of a building or building part are expressed by the class RoofSurface. Secondary parts of a roof with a specific semantic meaning like dormers or chimneys should be modelled as BuildingInstallation.

6.1.7. ClosureSurfaceType, ClosureSurface

NOTE insert ClosureSurfaceType, ClosureSurface UML

An opening in a building not filled by a door or window can be sealed by a virtual surface called ClosureSurface (cf. chapter 6.4). Hence, buildings with open sides like a barn or a hangar, can be virtually closed in order to be able to compute their volume. ClosureSurfaces are also used in the interior building model. If two rooms with a different function (e.g. kitchen and living room) are directly connected without a separating door, a ClosureSur-face should be used to separate or connect the volumes of both rooms.

6.1.8. FloorSurfaceType, FloorSurface

NOTE insert FloorSurfaceType, FloorSurface UML

The class FloorSurface must only be used in the LOD4 interior building model for modelling the floor of a room.

6.1.9. InteriorWallSurfaceType, InteriorWallSurface

NOTE insert InteriorWallSurfaceType, InteriorWallSurface UML

The class InteriorWallSurface must only be used in the LOD4 interior building model for modelling the visible surfaces of the room walls.

6.1.10. CeilingSurfaceType, CeilingSurface

NOTE Insert CeilingSurfaceType, CeilingSurface UML

The class CeilingSurface must only be used in the LOD4 interior building model for modelling the ceiling of a room.

6.2. Openings

6.2.1. AbstractOpeningType, _Opening

NOTE insert AbstractOpeningType, _Opening UML

The class _Opening is the abstract base class for semantically describing openings like doors or windows in outer or inner boundary surfaces like walls and roofs. Openings only exist in models of LOD3 or LOD4. Each _Opening is associated with a gml:MultiSurface geometry. Alternatively, the geometry may be given as Implic-itGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

6.2.2. WindowType, Window

NOTE insert WindowType, Window UML

The class Window is used for modelling windows in the exterior shell of a building, or hatches between adjacent rooms. The formal difference between the classes Window and Door is that – in normal cases – Windows are not specifically intended for the transit of people or vehicles.

6.2.3. DoorType, Door

NOTE insert DoorType, Door UML

The class Door is used for modelling doors in the exterior shell of a building, or between adjacent rooms. Doors can be used by people to enter or leave a building or room. In contrast to a ClosureSurface a door may be closed, blocking the transit of people. A Door may be assigned zero or more addresses. The corresponding Address-PropertyType is defined within the CityGML core module (cf. chapter 10.1.4) .

6.3. Building Interior

6.3.1. RoomType, Room

NOTE insert RoomType, Room UML

A Room is a semantic object for modelling the free space inside a building and should be uniquely related to exactly one building or building part object. It should be closed (if necessary by using ClosureSurfaces) and the geometry normally will be described by a solid (lod4Solid). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a MultiSurface (lod4MultiSurface). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when Room surfaces should be assigned Appearances. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised BoundarySurfaces (FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface cf. chapter 10.3.3).

A special task is the modelling of passages between adjacent rooms. The room solids are topologically connected by the surfaces representing hatches, doors or closure surfaces that seal open doorways. Rooms are defined as being adjacent, if they have common _Openings or ClosureSurfaces. The surface that represents the opening geometrically is part of the boundaries of the solids of both rooms, or the opening is referenced by both rooms on the semantic level. This adjacency implies an accessibility graph, which can be employed to determine the spread of e.g. smoke or gas, but which can also be used to compute escape routes using classical shortest path algorithms (see Fig. 33).

NOTE insert Fig 33

6.3.2. BuildingFurnitureType, BuildingFurniture

NOTE insert BuildingFurnitureType, BuildingFurniture UML

Rooms may have BuildingFurnitures and IntBuildingInstallations. A BuildingFurniture is a movable part of a room, such as a chair or furniture. A BuildingFurniture object should be uniquely related to exactly one room object. Its geometry may be represented by an explicit geometry or an

ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype building furniture is stored only once in a local coordi-nate system and referenced by other building furniture features (see chapter 8.2).

6.3.3. IntBuildingInstallationType, IntBuildingInstallation

NOTE insert IntBuildingInstallationType, IntBuildingInstallation UML

An IntBuildingInstallation is an object inside a building with a specialised function or semantic meaning. In contrast to BuildingFurniture, IntBuildingInstallations are permanently attached to the building structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class IntBuild-ingInstallation can either be associated with a room (class Room), or with the complete building / building part (class _AbstractBuilding, cf. chapter 10.3.1). However, they should be uniquely related to exactly one room or one building / building part object. An IntBuildingInstallation optionally has attributes class, function and usage. The attribute class, which can only occur once, represents a general classification of the internal building component. With the attributes function and usage, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an IntBuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype interior building installation is stored only once in a local coordi-nate system and referenced by other interior building installation features (see chapter 8.2). The visible surfaces of an interior building installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3).

6.4. Modelling building storeys using CityObjectGroups

CityGML does currently not provide a specific concept for the representation of storeys as it is available in the AEC/FM standard IFC (IAI 2006). However, a storey can be represented as an explicit aggregation of all build-ing features on a certain height level using CityGML's notion of CityObjectGroups (cf. chapter 10.11). This would include Rooms, Doors, Windows, IntBuildingInstallations and BuildingFurniture. If thematic surfaces like walls and interior walls should also be associated to a specific storey, this might require the vertical fragmenta-tion of these surfaces (one per storey), as in virtual 3D city models they typically span the whole façade.

In order to model building storeys with CityGML's generic grouping concept, a nested hierarchy of CityObject-Group objects has to be used. In a first step, all semantic objects belonging to a specific storey are grouped. The attributes of the corresponding CityObjectGroup object are set as follows:

- The class attribute shall be assigned the value "building separation".
- The function attribute shall be assigned the value "lodXStorey" with X between 1 and 4 in order to de-note that this group represents a storey wrt. a specific LOD.
- The storey name or number can be stored in the gml:name property. The storey number attribute shall be assigned the value "storeyNo_X" with decimal number X in order to denote that this group repre-sents a storey wrt. a specific number.

In a second step, the CityObjectGroup objects representing different storeys are grouped themselves. By using the generic aggregation concept of CityObjectGroup, the “storeys group” is associated with the corresponding Building or BuildingPart object. The class attribute of the storeys group shall be assigned the value “building storeys”.

6.5. Examples

The LOD1 model of the Campus North of the Karlruhe Institute of Technology (KIT) shown in Fig. 34 consists of 596 buildings and 187 building parts. The footprint geometries of the buildings are taken from a cadastral information system and extruded by a given height. Buildings with a unique identifier and a single height value are modeled as one building (bldg:Building). Buildings having a unique identifier but different height values are modeled as one building (bldg:Building) with one or more building parts (bldg:BuildingPart). Both buildings and building parts have solid geometries and their height values are additionally represented as thematic attribute (bldg:measuredHeight). Fig. 34 shows an aerial photograph of the KIT Campus North (left) and the CityGML LOD1 model (right).

NOTE insert Fig 34

An example for a fully textured LOD2 building model is given in Fig. 35 which shows the Bernhardus church located in the city of Karlsruhe, Germany. On the left side of Fig. 35, a photograph of the church in real world is shown whereas the CityGML building model of the church with photorealistic textures is illustrated on the right. The model is bounded by a ground surface, several wall and roof surfaces. The railing above the church clock is modeled as a building installation (BuildingInstallation).

NOTE insert Fig 35

The model shown in Fig. 36 was derived from a 3D CAD model generated during the planning phase of the building. On the left side of Fig. 36, the building is shown whereas on the right side the LOD3 model is present-ed. The building itself is bounded by wall surfaces, roof surfaces and a ground surface. Doors and windows are modeled including reveals. According to the cadaster data, the car port next to the building is not part of the building. Therefore the car port, the balcony and the chimney are modeled as building installations (BuildingIn-stallation). The model also contains the terrain intersection curve (lod3TerrainIntersection) as planned by the architect.

In order to determine the volume of the building, the geometries of all boundary surfaces, including doors and windows, are referenced by the building solid (lod3Solid) using the XLink mechanism. Consequently, the roof surfaces are split into surfaces representing the roof itself and surfaces representing the roof overhangs.

NOTE insert fig 36

Tunnel Model

The tunnel model is closely related to the building model. It supports the representation of thematic and spatial aspects of tunnels and tunnel parts in four levels of detail, LOD1 to LOD4. The tunnel model of CityGML is defined by the thematic extension module Tunnel (cf. chapter 7). Fig. 37 provides examples of tunnel models for each LOD.

NOTE insert Fig 37

The UML diagram of the tunnel model is shown in Fig. 38. The XML schema definition is attached in annex A.11. The pivotal class of the model is _AbstractTunnel, which is a subclass of the thematic class _Site (and transitively of the root class _CityObject). _AbstractTunnel is specialized either to a Tunnel or to a TunnelPart. Since an _AbstractTunnel consists of TunnelParts, which again are _AbstractTunnels, an aggregation hierarchy of arbitrary depth may be realized. As subclass of the root class _CityObject, an _AbstractTunnel inherits all properties from _CityObject like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences (cf. chapter 6.7). Further properties not explicitly covered by _AbstractTunnel may be modelled as generic attributes provided by the CityGML Generics module (cf. chapter 10.12) or using the CityGML Application Domain Extension mechanism (cf. chapter 10.13).

Both classes Tunnel and TunnelPart inherit the attributes of _AbstractTunnel: the class of the tunnel, the func-tion, the usage, the year of construction and the year of demolition. In contrast to _AbstractBuilding, Address features cannot be assigned to _AbstractTunnel.

NOTE insert Fig 38

The geometric representation and semantic structure of an _AbstractTunnel is shown in Fig. 38. The model is successively refined from LOD1 to LOD4. Therefore, not all components of a tunnel model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the proposed minimum acquisition criteria for each LOD (cf. chapter 6.2). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

Similar to the building and brigde models (cf. chapters 10.3 and 10.5), only the outer shell of a tunnel is repre-sented in LOD1 – 3, which is composed of the tunnel’s boundary surfaces to the surrounding earth, water, or outdoor air. The interior of a tunnel may only be modeled in LOD4. Although the interior built environment is especially relevant for subsurface objects like tunnels or underground buildings, CityGML employs a consistent LOD concept for all thematic modules. If, in contrast, the representation of the interior of subsurface objects would be possible in all LODs, the LOD concept for subsurface objects would have to substantially differ from the LOD concept for aboveground objects. This would require the precise definition of a “transition surface” which delimits the scope of both LOD concepts. Furthermore, features being partially above and below ground would have to be split into an above-ground part (modeled according to the aboveground LOD concept) and a subsurface part (modeled according to the subsurface LOD concept). However, such a splitting violates the CityGML concept of unity of features and would not be feasible in many cases where the transition between above and below ground is often not precisely known or depends on (the LOD of) the terrain model. Hence, CityGML applies a single and consistent LOD concept to both aboveground and subsurface objects. As a conse-quence, penetrations between a tunnel and objects inside this tunnel (e.g., roads and railways) may occur in LOD1 – 3.

In LOD1, a tunnel model consists of a geometric representation of the tunnel volume. Optionally, a MultiCurve representing the TerrainIntersectionCurve (cf. chapter 6.5) can be specified. The geometric representation is refined in LOD2 by additional MultiSurface and MultiCurve geometries.

In LOD2 and higher LODs the outer structure of a tunnel can also be differentiated semantically by the classes _BoundarySurface and TunnelInstallation. A boundary surface is a part of the tunnel's exterior shell with a special function like wall (WallSurface), roof (RoofSurface), ground plate (GroundSurface), outer floor (Outer-FloorSurface), outer ceiling (OuterCeilingSurface) or ClosureSurface. The TunnelInstallation class is used for tunnel elements like outer stairs, strongly affecting the outer appearance of a tunnel. A TunnelInstallation may have the attributes class, function and usage (see Fig. 38).

In LOD3, the openings in _BoundarySurface objects (doors and windows) can be represented as thematic ob-jects.

In LOD4, the highest level of resolution, also the interior of a tunnel, composed of several hollow spaces, is represented in the tunnel model by the class HollowSpace. This enlargement allows a virtual accessibility of tunnels, e.g. for driving through a tunnel, for simulating disaster management or for presenting the light illumi-nation within a tunnel. The aggregation of hollow spaces according to arbitrary, user defined criteria (e.g. for defining the hollow spaces corresponding to horizontal or vertical sections) is achieved by employing the general grouping concept provided by CityGML (cf. chapter 10.11). Interior installations of a tunnel, i.e. objects within a tunnel which (in contrast to furniture) cannot be moved, are represented by the class IntTunnelInstallation. If an installation is attached to a specific hollow space (e.g. lamps, ventilator), they are associated with the Hol-lowSpace class, otherwise (e.g. pipes) with _AbstractTunnel. A HollowSpace may have the attributes class, function and usage whose possible values can be enumerated in code lists (chapter 10.4.7, Annex C). The class attribute allows a general classification of hollow spaces, e.g. commercial or private rooms, and occurs only once. The function attribute is intended to express the main purpose of the hollow space, e.g. control area, installation space, storage space. The attribute usage can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

The visible surface of a hollow space is represented geometrically as a Solid or MultiSurface. Semantically, the surface can be structured into specialised _BoundarySurfaces, representing floor (FloorSurface), ceiling (Ceil-ingSurface), and interior walls (InteriorWallSurface). Hollow space furniture, like movable equipment in control areas, can be represented in the CityGML tunnel model with the class TunnelFurniture. A TunnelFurniture may have the attributes class, function and usage.

Tunnel and Tunnel Part

=====TunnelType, Tunnel

NOTE insert TunnelType, Tunnel UML

The Tunnel class is one of the two subclasses of _AbstractTunnel. If a tunnel only consists of one (homogene-ous) part, this class shall be used. A tunnel composed of structural segments, for example tunnel entrance and subway, has to be separated into one tunnel having one or more additional TunnelPart (see Fig. 39). The geome-try and non-spatial properties of the central part of the tunnel should be represented in the aggregating Tunnel feature.

TunnelPartType, TunnelPart

NOTE insert TunnelPartType, TunnelPart UML

If sections of a tunnel differ in geometry and / or attributes, the tunnel can be separated into parts (see Fig. 39). Like Tunnel, the class TunnelPart is derived from _AbstractTunnel and inherites all attributes of _AbstractTunnel. A TunnelPart object should be uniquely related to exactly one tunnel or tunnel part object.

NOTE insert Fig 39

AbstractTunnelType, _AbstractTunnel

NOTE insert AbstractTunnelType, _AbstractTunnel UML

The abstract class _AbstractTunnel contains properties for tunnel attributes, purely geometric representations, and geometric/semantic representations of the tunnel or tunnel part in different levels of detail. The attributes describe:

1. The classification of the tunnel or tunnel part (class), the different functions (function), and the usage (us-age). The type of these attributes is gml:CodeType and the values can be specified in separate code lists.
2. The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the tunnel or tunnel part. The yearOfConstruction is the year of completion of the tunnel. The yearOfDemolition is the year when the demolition of the tunnel was completed. The date (year) refer to real world time (e.g. 2011).

Spanning the different levels of detail, the tunnel model differs in the complexity and granularity of the geomet-ric representation and the thematic structuring of the model into components with a special semantic meaning. This is illustrated in Fig. 40, showing the same tunnel in four different LODs. Some properties of the class _AbstractTunnel are also associated with certain LODs.

NOTE insert Fig 40

Tab. 6 shows the correspondence of the different geometric and semantic themes of the tunnel model to LODs. In each LOD, the volume of a tunnel can be expressed by a gml:Solid geometry and/or a gml:MultiSurface geometry. The definition of a 3D Terrain Intersection Curve (TIC), used to integrate tunnels from different sources with the Digital Terrain Model, is also possible in all LODs. The TIC can – but does not have to – build closed rings around the tunnel or tunnel parts.

Table 2. Semantic themes of the class _AbstractTunnel

Geometric / semantic theme	Property type	LOD1	LOD2	LOD3	LOD4
Building footprint and roof edge	gml:MultiSurfaceType	•			
	Volume part of the tunnel shell	gml:Solid Type	•	•	•

Geometric / semantic theme	Property type	LOD1	LOD2	LOD3	LOD4
•	Surface part of the tunnel shell	gml:MultiSurfaceType	•	•	•
•	Terrain intersection curve	gml:MultiCurveType	•	•	•
•	Curve part of the tunnel shell	gml:MultiCurveType		•	•
•	Tunnel parts	TunnelPartType	•	•	•
•	Boundary surfaces (chapter 10.4.3)	AbstractBoundarySurfaceType		•	•
•	Outer tunnel installations (chapter 10.4.2)	TunnelInstallationType		•	•
•	Openings	AbstractOpeningType			•
•	Hollow spaces (chapter 10.4.5)	HollowSpaceType			
•	Interior tunnel installations	IntTunnelInstallationType			

Outer Tunnel Installations

TunnelInstallationType, TunnelInstallation

A TunnelInstallation is an outer component of a tunnel which has not the significance of a TunnelPart, but which strongly affects the outer characteristic of the tunnel, for examples stairs. A TunnelInstallation optionally has attributes class, function and usage. The attribute class - which can only occur once - represents a general classification of the installation. With the attributes function and usage, nominal and real functions of a tunnel installation can be described. For all three attributes the list of feasible values can specified in a code list. For the geometrical representation of a TunnelInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype tunnel installation is stored only once in a local coordinate system and referenced by other tunnel installation features (see chapter 8.2). The visible surfaces of a tunnel installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3). A TunnelInstalla-tion object should be uniquely related to exactly one tunnel or tunnel part object.

Boundary surfaces

NOTE insert Boundary surfaces UML

_BoundarySurface is the abstract base class for several thematic classes, structuring the exterior shell of a tunnel as well as the visible surface of hollow spaces and both outer and interior tunnel installations. It is a subclass of _CityObject and thus inherits all properties like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences. From _BoundarySurface, the thematic classes RoofSur-face, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface, ClosureSurface, FloorSurface, InteriorWallSurface, and CeilingSurface are derived. The thematic classification of tunnel surfaces is illustrated in Fig. 41 for different types of tunnel cross sections and are specified below.

NOTE insert Fig 41

For each LOD between 2 and 4, the geometry of a _BoundarySurface may be defined by a different gml:MultiSurface geometry. Starting from LOD3, a _BoundarySurface may contain _Openings (cf. chapter 10.4.4) like doors and windows. If the geometric location of openings topologically lies within a surface compo-nent (e.g. gml:Polygon) of the gml:MultiSurface geometry, these openings must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. Accord-ing to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface’s normal vector). If such an opening is sealed by a Door or a Window, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an opening belong to the relevant adjacent _BoundarySurface. If, for example a door seals the opening, the embrasure surface on the one side of the door belongs to the InteriorWallSurface and on the other side to the WallSurface (cf. right part of Fig. 32 for the same situation in a building model).

GroundSurfaceType, GroundSurface

NOTE insert GroundSurfaceType, GroundSurface UML

The ground plate of a tunnel or tunnel part is modelled by the class GroundSurface. Usually a GroundSurface is a boundary surface between the tunnel and the surrounding earth (soil, rock etc.) or water.

OuterCeilingSurfaceType, OuterCeilingSurface

NOTE insert OuterCeilingSurfaceType, OuterCeilingSurface UML

A mostly horizontal surface belonging to the outer tunnel shell and with the orientation pointing downwards can be modeled as an OuterCeilingSurface. Examples are the visible part of an avalanche protector or the boundary surface between the tunnel and the surrounding earth or water.

WallSurfaceType, WallSurface

NOTE insert WallSurfaceType, WallSurface UML

All parts of the tunnel facade belonging to the outer tunnel shell can be modelled by the class WallSurface. Usually a WallSurface is a boundary surface between the tunnel and the surrounding earth (soil, rock etc.) or water.

OuterFloorSurfaceType, OuterFloorSurface

NOTE insert OuterFloorSurfaceType, OuterFloorSurface UML

A mostly horizontal surface belonging to the outer tunnel shell and with the orientation pointing upwards can be modeled as an OuterFloorSurface.

RoofSurfaceType, RoofSurface

NOTE insert RoofSurfaceType, RoofSurface UML

Boundary surfaces belonging to the outer tunnel shell and with the main purpose to protect the tunnel from above are expressed by the class RoofSurface. The orientation of these boundaries is mainly pointing upwards.

ClosureSurfaceType, ClosureSurface

NOTE insert ClosureSurfaceType, ClosureSurface UML

Openings in tunnels or hollow spaces not filled by a door or a window can be sealed by a virtual surface called ClosureSurface (cf. chapter 6.4). For example, the doorways of tunnels can be modelled as ClosureSurface.

FloorSurfaceType, FloorSurface

NOTE insert FloorSurfaceType, FloorSurface UML

The class FloorSurface must only be used in the LOD4 interior tunnel model for modelling the floor of hollow spaces.

InteriorWallSurfaceType, InteriorWallSurface

NOTE insert InteriorWallSurfaceType, InteriorWallSurface UML

The class InteriorWallSurface is only allowed to be used in the LOD4 interior tunnel model for modelling the visible wall surfaces of hollow spaces.

CeilingSurfaceType, CeilingSurface

NOTE insert CeilingSurfaceType, CeilingSurface UML

The class CeilingSurface is only allowed to be used in the LOD4 interior tunnel model for modelling the ceiling of hollow spaces.

Openings

AbstractOpeningType, _Opening

NOTE insert AbstractOpeningType, _Opening UML

The class _Opening is the abstract base class for semantically describing openings like doors or windows in outer and inner boundary surfaces. Openings only exist in models of LOD3 or LOD4. Each _Opening is associ-ated with a gml:MultiSurface geometry. Alternatively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

NOTE insert WindowType, Window UML

The class Window is used for modelling windows in the in the exterior shell of a tunnel and in hollow spaces, or hatches between adjacent hollow spaces. The formal difference between the classes Window and Door is that – in normal cases – Windows are not specifically intended for the transit of people or vehicles.

DoorType, Door

NOTE insert DoorType, Door UML

The class Door is used for modelling doors in the exterior shell of a tunnel, or between adjacent hollow spaces. Doors can be used by people to enter or leave a tunnel or ahollow space. In contrast to a ClosureSurface a door may be closed, blocking the transit of people or vehicles.

Tunnel Interior

HollowSpaceType, HollowSpace

NOTE insert HollowSpaceType, HollowSpace UML

A HollowSpace is a semantic object for modelling the free space inside a tunnel and should be uniquely related to exactly one tunnel or tunnel part object. It should be closed (if necessary by using ClosureSurface) and the geometry normally will be described by a solid (lod4Solid). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a MultiSurface (lod4MultiSurface). The surface normals of the outer shell of a GML solid must point outwards. This is important if appearences should be assigned to HollowSpace surfaces. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the hollow space.

In addition to the geometrical representation, different parts of the visible surface of a hollow space can be modelled by specialised boundary surfaces (FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSur-face, cf. chapter 10.4.3).

TunnelFurnitureType, TunnelFurniture

NOTE insert TunnelFurnitureType, TunnelFurniture UML

Hollow spaces may have TunnelFurniture. A TunnelFurniture is a movable part of a hollow space. A Tunnel-Furniture object should be uniquely related to exactly one hollow space. Its geometry may be represented by an explicit geometry or an ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype tunnel furniture is stored only once in a local coordinate system and referenced by other tunnel furni-ture features (see chapter 8.2).

IntTunnelInstallationType, IntTunnelInstallation

NOTE insert IntTunnelInstallationType, IntTunnelInstallation UML

An IntTunnelInstallation is an object inside a tunnel with a specialized function or semantic meaning. In contrast to TunnelFurniture, objects of the class IntTunnelInstallation are permanently attached to the tunnel structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class IntTunnelInstallation can either be associated with a hollow space (class HollowSpace), or with the complete tunnel or tunnel part (class _AbstractTunnel, see chapter 10.4.1). However, they should be uniquely related to exactly one hollow space or one tunnel / tunnel part object. An IntTunnelInstallation optionally has the attributes class, function and usage. The attribute class, which can only occur once, represents a general classification of the internal tunnel component. With the attributes function and usage, nominal and real functions of a tunnel installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an IntTunnelInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype interior tunnel installation is stored only once in a local coordinate system and referenced by other interior tunnel installation features (see chapter 8.2). The visible surfaces of an interior tunnel installation can be semantically classified using the concept of boundary surfaces (cf. 10.4.3).

Examples

The example in Fig. 42 shows a pedestrian underpass in the city centre of Karlsruhe, Germany. On the left side of Fig. 42, a photo illustrates the real world situation. Both entrances of the underpass are marked in the photo by dashed rectangles. On the right side of the figure, the CityGML tunnel model is shown. The terrain surrounding the tunnel has been virtually cut out of model in order to visualize the entire tunnel with its subsurface body. The same underpass is illustrated in Fig. 43 from a different perspective. The camera is positioned in front of the left entrance (black dashed rectangle in Fig. 42) and pointing in the direction of the right entrance (white dashed rectangle in Fig. 42). On the right side of Fig. 43, the tunnel model is shown from the same perspective. Again holes are cut in the terrain surface in order to make the subsurface part of the tunnel visible. An LOD1 representation of the nearby buildings is shown in the background of the model.

NOTE insert Fig 42

NOTE insert Fig 43

The model is subdivided into one Tunnel (the actual underpass) and two TunnelParts (both entrances). The tunnel and tunnel parts are bounded by GroundSurface, WallSurface, RoofSurface. ClosureSurface objects are used to virtually seal the tunnel entrances. For safety reasons each of the two entrances has railings which are modeled as TunnelInstallation. Due to the high geometrical accuracy and the semantic richness, the model is classified as LOD3.

Bridge Package

The bridge model allows for the representation of the thematic, spatial and visual aspects of bridges and bridge parts in four levels of detail, LOD 1 – 4. The bridge model of CityGML is defined by the thematic extension module Bridge (cf. chapter 7). Fig. 44 illustrates examples of bridge models in all LODs.

NOTE insert Fig 44

The bridge model was developed in analogy to the building model (cf. chapter 10.3) with regard to structure and attributes. The UML diagram of the bridge model is depicted in Fig. 45.

NOTE insert fig 45 UML Model

A (movable or unmovable) bridge is represented by an object of the class Bridge. This class inherits its attributes and relations from the abstract base class _AbstractBridge. The spatial properties are defined by a solid for each of the four LODs (relations lod1Solid to lod4Solid). In analogy to the building model, the semantical as well as the geometrical richness increases from LOD1 (blocks model) to LOD3 (architectural model). Simple examples of bridges in each of those LODs are depicted in Fig. 46. Interior structures like rooms are dedicated to LOD4. To cover the case of bridge models where the topology does not satisfy the properties of a solid (essentially water tightness), a multi surface representation is allowed (lod1MultiSurface to lod4MultiSurface). The line where the bridge touches the terrain surface is represented by a terrain intersection curve, which is provided for each LOD (relations lod1TerrainIntersection to lod4TerrainIntersection). In addition to the solid representation of a bridge, linear characteristics like ropes or antennas can be specified geometrically by the lod1MultiCurve to lod4MultiCurve relations. If those characteristics shall be represented semantically, the features BridgeInstalla-tion or BridgeConstructionElement can be used (see section 10.5.2). All relations to semantic objects and geo-metric properties are listed in Tab. 7.

NOTE insert Fig 46

The semantic attributes of an _AbstractBridge are class, function, usage and is_movable. The attribute class is used to classify bridges, e.g. to distinguish different construction types (cf. Fig. 48). The attribute function allows representing the utilization of the bridge independently of the construction. Possible values may be railway bridge, roadway bridge, pedestrian bridge, aqueduct, etc. The option to denote a usage which is divergent to one of the primary functions of the bridge (function) is given by the attribute usage. The type of these attributes is gml:CodeType, the values of which can be defined in code lists. The name of the bridge can be represented by the gml:name attribute, which is inherited from the base class gml:_GML via the classes gml:_Feature, _CityObject, and _Site. Each Bridge or BridgePart feature may be assigned zero or more addresses using the address property. The corresponding AddressPropertyType is defined within the CityGML core module (cf. chapter 10.1.4).

Table 3. Semantic themes of the class _AbstractBridge

Geometric / semantic theme	Property type	LOD1	LOD2	LOD3	LOD4
Volume part of the bridge shell	gml:SolidType	•	•	•	•
Surface part of the bridge shell	gml:MultiSurfaceType	•	•	•	•
Terrain intersection curve	gml:MultiCurveType	•	•	•	•
Curve part of the bridge shell	gml:MultiCurveType		•	•	•
Bridge parts (chapter 10.5.1)	BridgePartType	•	•	•	•
Boundary surfaces (chapter 10.5.3)	AbstractBoundarySurfaceType		•	•	•
Outer bridge installations (chapter 10.5.2)	BridgeInstallationType		•	•	•
Bridge construction elements (chapter 10.5.2)	BridgeConstruction-ElementType	•	•	•	•
Openings (chapter 10.5.4)	AbstractOpeningType			•	•
Bridge rooms (chapter 10.5.5)	BridgeRoomType				•
Interior bridge installations	IntBridgeInstallation Type				•

The boolean attribute is_movable is defined to specify whether a bridge is movable or not. The modeling of the geometric aspects of the movement is delayed to later versions of this standard. Some types of movable bridges are depicted in Fig. 47.

NOTE insert Fig 47

NOTE insert Fig 48

Bridge and bridge part

BridgeType, Bridge

NOTE insert BridgeType, Bridge UML

BridgePartType, BridgePart

NOTE insert BridgePartType, BridgePart UML

If some parts of a bridge differ from the remaining bridge with regard to attribute values or if parts like ramps can be identified as objects of their own, those parts can be represented as BridgePart. A bridge can consist of multiple BridgeParts. Like Bridge, BridgePart is a subclass of _AbstractBridge and hence, has the same attrib-utes and relations. The relation consistOfBridgePart represents the aggregation hierarchy between a Bridge (or a BridgePart) and it’s BridgeParts. By this means, an aggregation hierarchy of arbitrary depth can be modeled. Each BridgePart belongs to exactly one Bridge (or BridgePart). Similar to the building model, the aggregation structure of a bridge forms a tree. A simple example for a bridge with parts is a twin bridge. Another example is presented in chapter 10.5.6.

AbstractBridgeType, _AbstractBridge

NOTE insert AbstractBridgeType, _AbstractBridge UML

The abstract class _AbstractBridge is the base class of Bridges and BridgeParts. It contains properties for bridge attributes, purely geometric representations, and geometric/semantic representations of the bridge or bridge part in different levels of detail. The attributes describe:

1. The classification of the bridge or bridge part (class), the different intended usages (function), and the different actual usages (usage). The permitted values for these property types can be specified in code lists.
2. The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the bridge or bridge part. These attributes can be used to describe the chronology of the bridge development within a city model. The points of time refer to real world time.
3. Whether the bridge is movable is specified by the Boolean attribute isMovable.

Bridge construction elements and bridge installations

BridgeConstructionElementType, BridgeConstructionElement

NOTE insert BridgeConstructionElementType, BridgeConstructionElement UML

BridgeInstallationType, BridgeInstallation

NOTE insert BridgeInstallationType, BridgeInstallation UML

Bridge elements which do not have the size, significance or meaning of a BridgePart can be modelled either as BridgeConstructionElement or as BridgeInstallation. Elements which are essential from a structural point of view are modelled as BridgeConstructionElement, for example structural elements like pylons, anchorages etc. (cf. Fig. 49). A general classification as well as the intended and actual function of the construction element are represented by the attributes class, function, and usage. The geometry of a BridgeConstructionElement, which may be present in LOD1 to LOD4, is gml:_Geometry. Alternatively, the geometry may be given as ImplicitGe-ometry object. Following the concept of ImplicitGeometry the geometry of a prototype bridge construction element is stored only once in a local coordinate system and referenced by other bridge construction element features (cf. chapter 8.2). The visible surfaces of a bridge construction element can be semantically classified using the concept of boundary surfaces (cf. chapter 10.5.3).

Whereas a BridgeConstructionElement has structural relevance, a BridgeInstallation represents an element of the bridge which can be eliminated without collapsing of the bridge (e.g. stairway, antenna, railing). BridgeInstalla-tions occur in LOD 2 to 4 only and are geometrically representated as gml:_Geometry. Again, the concept of ImplicitGeometry can be applied to BridgeInstallations alternatively, and their visible surfaces can be semanti-cally classified using the concept of boundary surfaces (cf. chapter 10.5.3). The class BridgeInstallation contains the semantic attributes class, function and usage. The attribute class gives a classification of installations of a bridge. With the attributes function and usage, nominal and real functions of the bridge installation can be described. The type of all attributes is gml:CodeType and their values can be defined in code lists.

NOTE insert Fig 49

Boundary surfaces

AbstractBoundarySurfaceType, _BoundarySurface

NOTE insert AbstractBoundarySurfaceType, _BoundarySurface UML

The thematic boundary surfaces of a bridge are defined in analogy to the building module. _BoundarySurface is the abstract base class for several thematic classes, structuring the exterior shell of a bridge as well as the visible surfaces of rooms, bridge construction elements and both outer and interior bridge installations. It is a subclass of _CityObject and thus inherits all properties like the GML3 standard feature properties (gml:name etc.) and the CityGML specific properties like ExternalReferences. From _BoundarySurface, the thematic classes RoofSur-face, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface, ClosureSurface, FloorSurface, InteriorWallSurface, and CeilingSurface are derived.

For each LOD between 2 and 4, the geometry of a _BoundarySurface may be defined by a different gml:MultiSurface geometry.

In LOD3 and LOD4, a _BoundarySurface may contain _Openings (cf. chapter 10.5.4) like doors and windows. If the geometric location of _Openings topologically lies within a surface component (e.g. gml:Polygon) of the gml:MultiSurface geometry, these _Openings must be represented as holes within that surface. A hole is repre-sented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface's normal vector). If such an opening is sealed by a Door, a Window, or a

ClosureSurface, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an Opening belong to the relevant adjacent _BoundarySurface. If, for example a door seals the Opening, the embrasure surface on the one side of the door belongs to the InteriorWallSurface and on the other side to the WallSurface.

Fig. 50 depicts a bridge with RoofSurfaces, WallSurfaces, OuterFloorSurfaces and OuterCeilingSurfaces. Besides Bridges and BridgeParts, BridgeConstructionElements, BridgeInstallations as well as IntBridgeInstalla-tions can be related to _BoundarySurface. _BoundarySurfaces occur in LOD2 to LOD4. In LOD3 and LOD4, such a surface may contain _Openings (see chapter 10.3.4) like doors and windows.

NOTE insert fig 50

GroundSurfaceType, GroundSurface

NOTE insert GroundSurfaceType, GroundSurface UML

The ground plate of a bridge or bridge part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the bridge’s footprint. However, the surface normal of the ground plate is point-ing downwards.

OuterCeilingSurfaceType, OuterCeilingSurface

Note: insert OuterCeilingSurfaceType, OuterCeilingSurface UML

A mostly horizontal surface belonging to the outer bridge shell and having the orientation pointing downwards can be modeled as an OuterCeilingSurface.

WallSurfaceType, WallSurface

NOTE insert WallSurfaceType, WallSurface UML

All parts of the bridge facade belonging to the outer bridge shell can be modelled by the class WallSurface

OuterFloorSurfaceType, OuterFloorSurface

NOTE insert OuterFloorSurfaceType, OuterFloorSurface UML

A mostly horizontal surface belonging to the outer bridge shell and with the orientation pointing upwards can be modeled as an OuterFloorSurface

RoofSurfaceType, RoofSurface

NOTE insert RoofSurfaceType, RoofSurface UML

The major roof parts of a bridge or bridge part are expressed by the class RoofSurface.

ClosureSurfaceType, ClosureSurface

NOTE insert ClosureSurfaceType, ClosureSurface UML

An opening in a bridge not filled by a door or window can be sealed by a virtual surface called ClosureSurface (cf. chapter 6.4). Hence, bridge with open sides can be virtually closed in order to be able to compute their volume. ClosureSurfaces are also used in the interior bridge model. If two rooms with a different are directly connected without a separating door, a ClosureSurface should be used to separate or connect the volumes of both rooms.

FloorSurfaceType, FloorSurface

NOTE insert FloorSurfaceType, FloorSurface UML

The class FloorSurface must only be used in the LOD4 interior bridge model for modelling the floor of a bridge room.

InteriorWallSurfaceType, InteriorWallSurface

NOTE insert InteriorWallSurfaceType, InteriorWallSurface UML

The class InteriorWallSurface must only be used in the LOD4 interior bridge model for modelling the visible surfaces of the bridge room walls.

CeilingSurfaceType, CeilingSurface

NOTE insert CeilingSurfaceType, CeilingSurface UML

The class CeilingSurface must only be used in the LOD4 interior bridge model for modelling the ceiling of a bridge room.

Openings

AbstractOpeningType, _Opening

NOTE insert AbstractOpeningType, _Opening UML

The class _Opening is the abstract base class for semantically describing openings like doors or windows in outer or inner boundary surfaces like walls and roofs. Openings only exist in models of LOD3 or LOD4. Each _Opening is associated with a gml:MultiSurface geometry. Alternatively, the geometry may be given as Implic-itGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

NOTE insert WindowType, Window UML

The class Window is used for modelling windows in the exterior shell of a bridge, or hatches between adjacent rooms. The formal difference between the classes Window and Door is that – in normal cases – Windows are not specifically intended for the transit of people or vehicles.

DoorType, Door

NOTE insert DoorType, Door UML

The class Door is used for modelling doors in the exterior shell of a bridge, or between adjacent rooms. Doors can be used by people to enter or leave a bridge or room. In contrast to a ClosureSurface a door may be closed, blocking the transit of people. A Door may be assigned zero or more addresses. The corresponding Address-PropertyType is defined within the CityGML core module (cf. chapter 10.1.4).

Bridge Interior

The classes BridgeRoom, IntBridgeInstallation and BridgeFurniture allow for the representation of the bridge interior. They are designed in analogy to the classes Room, IntBuildingInstallation and BuildingFurniture of the building module and share the same meaning. The bridge interior can only be modeled in LOD4.

BridgeRoomType, BridgeRoom

NOTE insert BridgeRoomType, BridgeRoom UML

A BridgeRoom is a semantic object for modelling the free space inside a bridge and should be uniquely related to exactly one bridge or bridge part object. It should be closed (if necessary by using ClosureSurfaces) and the geometry normally will be described by a solid (lod4Solid). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a MultiSurface (lod4MultiSurface). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when BridgeRoom surfaces should be assigned Appearances. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialised BoundarySurfaces (FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface; cf. chapter 10.5.3).

BridgeFurnitureType, BridgeFurniture

NOTE insert BridgeFurnitureType, BridgeFurniture UML

BridgeRooms may have BridgeFurnitures and IntBridgeInstallations. A BridgeFurniture is a movable part of a room, such as a chair or furniture. A BridgeFurniture object should be uniquely related to exactly one room object. Its geometry may be represented by an explicit geometry or an ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype bridge furniture is stored only once in a local coordi-nate system and referenced by other bridge furniture features (see chapter 8.2).

IntBridgeInstallationType, IntBridgeInstallation

NOTE insert IntBridgeInstallationType, IntBridgeInstallation UML

An IntBridgeInstallation is an object inside a bridge with a specialised function or semantic meaning. In contrast to BridgeFurniture, IntBridgeInstallations are permanently attached to the bridge structure and cannot be moved. Examples for IntBridgeInstallations are stairways, railings and heaters. Objects of the class IntBridgeInstallation can either be associated with a room (class BridgeRoom), or with the complete bridge / bridge part (class _AbstractBridge, cf. chapter 10.5.1). However, they should be uniquely related to exactly one room or one bridge / bridge part object. An IntBridgeInstallation optionally has attributes class, function and usage. The attribute class, which can only occur once, represents a general classification of the internal bridge component. With the attributes function and usage, nominal and real functions of a bridge installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an IntBridgeInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alter-natively, the geometry may be given as ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype interior bridge installation is stored only once in a local coordinate system and refer-enced by other interior bridge installation features (see chapter 8.2). The visible surfaces of an interior bridge installation can be semantically classified using the concept of boundary surfaces (cf. 10.5.3).

Examples

The bridge of Rees crossing the Rhine in Germany has three bridge parts which are separated by pylons. Fig. 51 (left) depicts the Rees bridge model containing one Bridge feature which consists of three BridgePart features. The pylons, which are structurally essential, are represented by BridgeConstructionElements. On the top of the pylons, four lamps are located which are modeled as BridgeInstallation features (cf. right part of Fig. 51).

NOTE insert Fig 51

In the following Fig. 52, the main part of the bridge of Rees is shown as photograph on the left side (source: Harald Halfpapp), and the corresponding part of the LOD2 bridge model is depicted on the the right side (source: District of Recklinghausen / KIT).

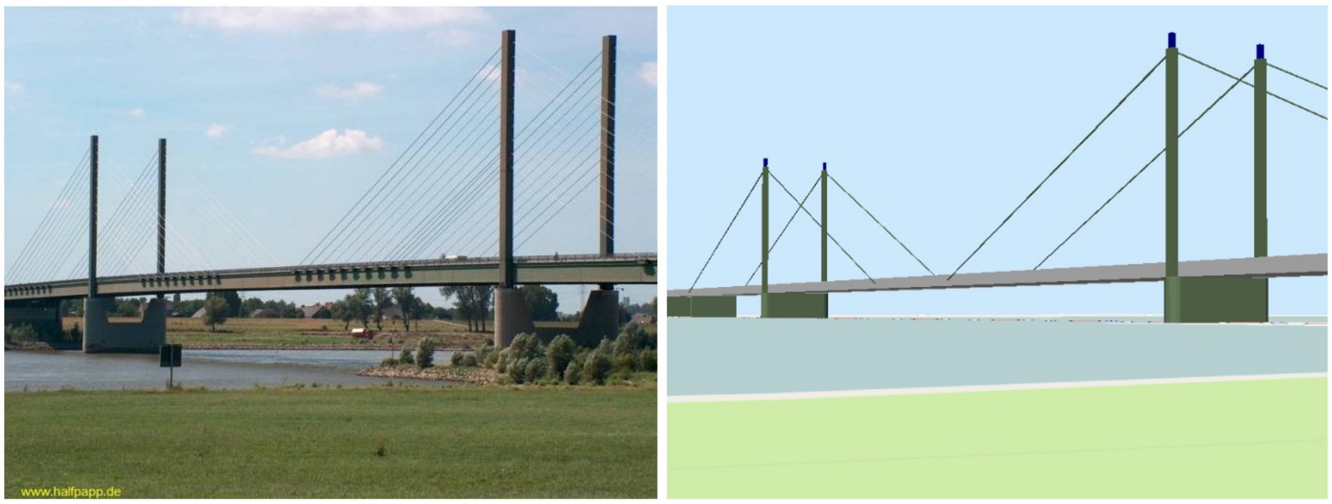

Figure 5. The bridge of Rees (left photo (source: Harald Halfpapp); right LOD2 model (source: District of Recklinghausen / KIT)).

There are two bridges crossing the river Rhine at Karlsruhe, Germany. The first one is a two track railway bridge constructed as a truss bridge (cf. Fig. 53 front). The second one is a four lane highway bridge constructed as a cable-stayed bridge (cf. Fig. 53 background).

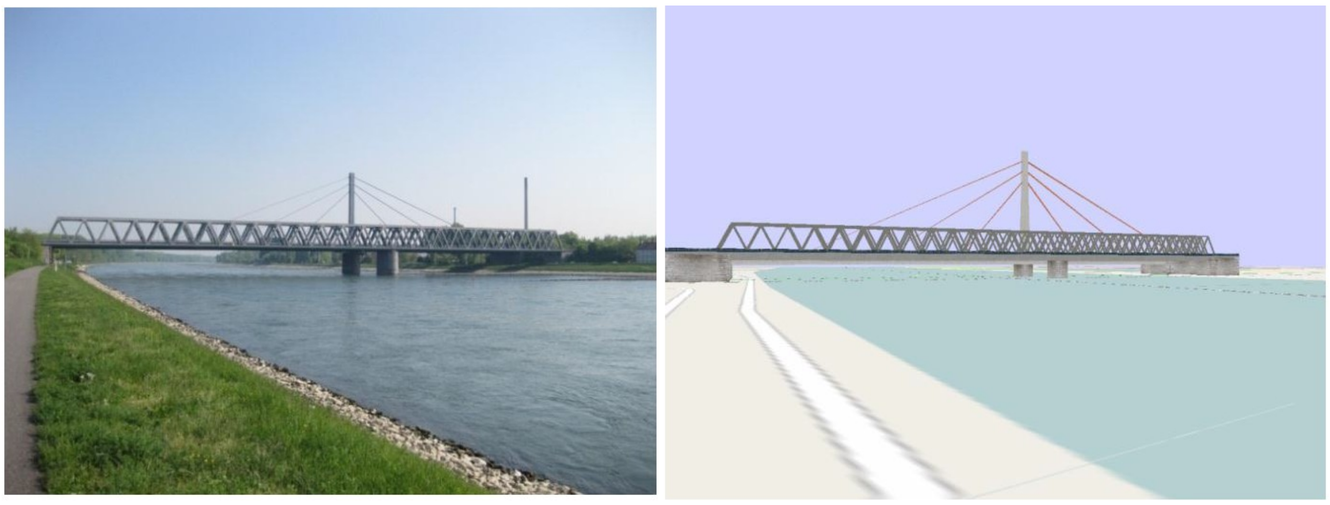

Figure 6. Bridge over the river Rhine at Karlsruhe (left a photo, right the 3D CityGML model) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).

In CityGML both bridges are modeled as single Bridge object with BridgeConstructionElements and BridgeIn-stallations. The construction elements of the cable stayed bridge are the footings on both river sides and in the middle of the river, as well as the cables and the pylon. The construction elements of the truss bridge are the footings and the truss itself. Both bridges have several railings which are modeled as BridgeInstallation. The bridge “Oberbaumbrücke” shown in Fig. 54 is located in the centre of Berlin crossing the river Spree and serves as example for bridges having interior rooms. The real-world bridge is depicted in the left part of Fig. 54, whereas the corresponding CityGML model is shown on the right. The outer geometry of the bridge is modeled as gml:MultiSurface element (lod4MultiSurface property) and is assigned photorealistic textures. Additionally, the interior rooms located in both bridge towers are represented as BridgeRoom objects with solid geometries (gml:Solid assigned through the lod4Solid property). Due to its geometric accuracy and the representation of the interior structures of both bridge towers, the model is classified as LOD4.

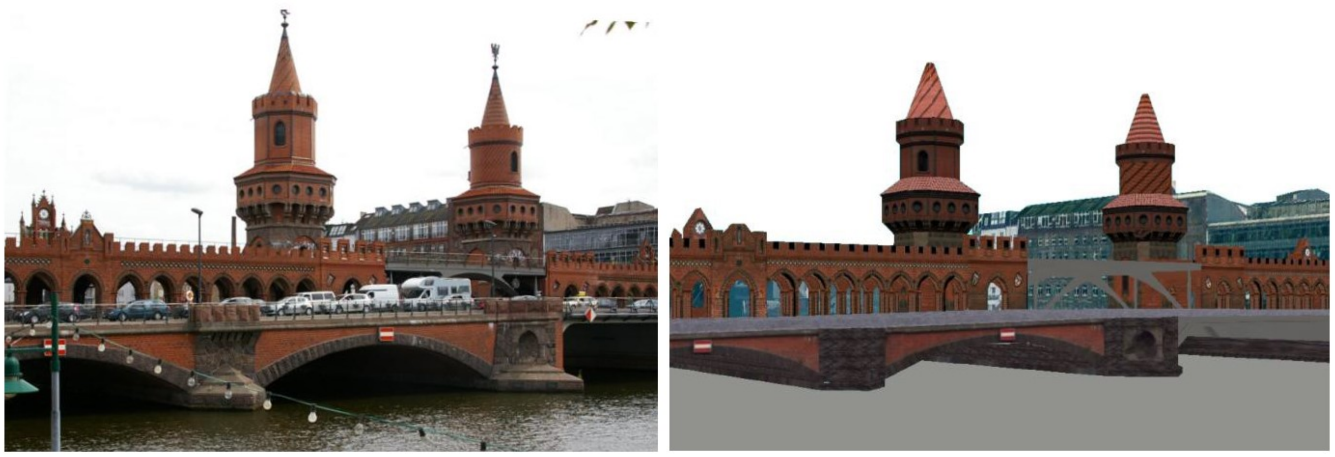

Figure 7. The bridge“Oberbaumbrücke” in Berlin represented as bridge model in LOD4 (left a photo, right the 3D CityGML model) (source: Berlin Senate of Business, Technology and Women; Business Location Center, Berlin; Technische Universität Berlin; Karlsruhe Institute of Technology (KIT)).

Water Bodies

Waters have always played an important role in urbanisation processes and cities were built preferably at rivers and places where landfall seemed to be easy. Obviously, water is essential for human alimentation and sanita-tion. Water bodies present the most economical way of transportation and are barriers at the same time, that avoid instant access to other locations. Bridging waterways caused the first efforts of construction and resulted in high-tech bridges of today. The landscapes of many cities are dominated by water, which directly relates to 3D city models. Furthermore, water bodies are important for urban life as subject of recreation and possible hazards as e.g. floods.

The distinct character of water bodies compared with the permanence of buildings, roadways, and terrain is considered in this thematic model. Water bodies are dynamic surfaces. Tides occur regularly, but irregular events predominate with respect to natural forces, for example flood events. The visible water surface changes in height and its covered area with the necessity to model its semantics and geometry distinct from adjacent objects like terrain or buildings.

This first modelling approach of water bodies fulfils the requirements of 3D city models. It does not inherit any hydrological or other dynamic aspects. In these terms it does not claim to be complete. However, the semantic and geometric description given here allows further enhancements of dynamics and conceptually different descriptions. The water bodies model of CityGML is embraced by the extension module WaterBody (cf. chapter 7).

The water bodies model represents the thematic aspects and three-dimensional geometry of rivers, canals, lakes, and basins. In the LOD 2-4 water bodies are bounded by distinct thematic surfaces. These surfaces are the obligatory WaterSurface, defined as the boundary between water and air, the optional WaterGroundSurface, defined as the boundary between water and underground (e.g. DTM or floor of a 3D basin object), and zero or more WaterClosureSurfaces, defined as virtual boundaries between different water bodies or between water and the end of a modelled region (see Fig. 55). A dynamic element may be the WaterSurface to represent temporarily changing situations of tidal flats.

NOTE insert Fig 55

The UML diagram of the water body model is depicted in Fig. 56, for the XML schema definition see below and annex A.13. Each WaterBody object may have the attributes class, function and usage whose possible values can be enumerated in code lists (cf. chapter 10.6.3 and annex C.9). The attribute class defines the classification of the object, e.g. lake, river, or fountain and can occur only once. The attribute function contains the purpose of the object like, for example national waterway or public swimming, while the attribute usage defines the actual usages, e.g. whether the water body is navigable. The latter two attributes can occur multiple times.

WaterBody is a subclass of _WaterObject and transitively of the root class _CityObject. The class _WaterObject may be differentiated in further subclasses of water objects in the future. The geometrical representation of the WaterBody varies through the different levels of detail. Since WaterBody is a subclass of _CityObject and hence a feature, it inherits the attribute gml:name. The WaterBody can be differentiated semantically by the class _WaterBoundarySurface. A _WaterBoundarySurface is a part of the water body's exterior shell with a special function like WaterSurface, WaterGroundSurface or WaterClosureSurface. As with any _CityObject, WaterBody objects as well as WaterSurface, WaterGroundSurface, and WaterClosureSurface may be assigned ExternalRef-erences (cf. chapter 6.7) and may be augmented by generic attributes using CityGML's Generics module (cf. chapter 10.12).

The optional attribute waterLevel of a WaterSurface can be used to describe the water level, for which the given 3D surface geometry was acquired. This is especially important when the water body is influenced by the tide. The allowed values can be defined in a corresponding code list.

NOTE insert Fig56 UML

Both LOD0 and LOD1 represent a low level of illustration and high grade of generalisation. Here the rivers are modelled as MultiCurve geometry and brooks are omitted. Seas, oceans and lakes with significant extent are represented as a MultiSurface (Fig. 56). Every WaterBody may be assigned a combination of geometries of different types. Linear water bodies are represented as a network of 3D curves. Each curve is composed of straight line segments, where the line orientation denotes the flow direction (water flows from the first point of a curve, e.g. a gml:LineString, to the last). Areal objects like lakes or seas are represented by 3D surface geome-tries of the water surface.

Starting from LOD1 water bodies may also be modelled as water filled volumes represented by Solids. If a water body is represented by a gml:Solid in LOD2 or higher, the surface geometries of the corresponding thematic WaterClosureSurface, WaterGroundSurface, and WaterSurface objects must coincide with the exterior shell of the gml:Solid. This can be ensured, if for each LOD X the respective lodXSolid representation (where X is between 2 and 4) does not redundantly define the geometry, but instead references the corresponding polygons (using GML3's XLink mechanism) of the lodXSurface elements (where X is between 2 and 4) of Water-ClosureSurface, WaterGroundSurface, and WaterSurface.

LOD2 to LOD4 demand a higher grade of detail and therefore any WaterBody can be outlined by thematic surfaces or a solid composed of the surrounding thematic surfaces. Every object of the class WaterSurface, WaterClosureSurface, and WaterGroundSurface must have at least one associated surface geometry. This means, that every WaterSurface, WaterClosureSurface, and WaterGroundSur-face feature within a CityGML instance document must contain at least one of the following properties: lod2Surface, lod3Surface, lod4Surface.

The water body model implicitly includes the concept of TerrainIntersectionCurves (TIC), e.g. to specify the exact intersection of the DTM with the 3D geometry of a WaterBody or to adjust a WaterBody or WaterSurface to the surrounding DTM (see chapter 6.5). The rings defining the WaterSurface polygons implicitly delineate the intersection of the water body with the terrain or basin.

Water Body

AbstractWaterObjectType, _WaterObject

NOTE insert AbstractWaterObjectType, _WaterObject UML

WaterBodyType, WaterBody

NOTE insert WaterBodyType, WaterBody UML

Boundary surfaces

With respect to different functions and characteristics three boundary classes for water are defined to build a solid or composite surface geometry (Fig. 55).

1. Boundary class “Air to Water”. The WaterSurface is mandatory to the model and usually is registered using photogrammetric analysis or mapping exploration. The representation may vary due to tidal flats or chang-ing water levels, which can be reflected by including different static water surfaces having different wa-terLevels (gml:CodeType), as for example highest flooding event, mean sea level, or minimum water level. This offers the opportunity to describe significant water surfaces due to levels that are important for certain representations e.g. in tidal zones.
2. Boundary class “Water to Ground”. The WaterGroundSurface may be known by sonar exploration or other depth measurements. Also part of the ground surface is the boundary “Water to Construction”. The ground surface might be identical to the underwater terrain model, but also describes the contour to other underwa-ter objects. The usefulness of this concept arises from the existence of water defence constructions like sluices, sills, flood barrage or tidal power stations. The use of WaterGroundSurface as boundary layer to man-made constructions is relevant in urban situations, where such objects may enclose the modeled water body completely, for example fountains and swimming pools. The WaterSurface objects together with the WaterGroundSurface objects enclose the WaterBody as a volume.
3. Boundary class “Water to Water”. The WaterClosureSurface is an optional feature that comes in use when the union of the WaterSurfaces and WaterGroundSurfaces of a water body does not define a closed volume. The WaterClosureSurface is then used to complete the enclosure of water volumes and to separate water volumes from those where only the surface is known. This might occur, where the cross section and ground surface of rivers is partly available during its course.

_WaterBoundarySurfaces should only be included as parts of corresponding WaterBody objects and should not be used as stand-alone objects within a CityGML model.

AbstractWaterBoundarySurfaceType, _WaterBoundarySurface

NOTE insert AbstractWaterBoundarySurfaceType, _WaterBoundarySurface UML

WaterSurfaceType, WaterSurface

NOTE insert WaterSurfaceType, WaterSurface UML

WaterGroundSurfaceType, WaterGroundSurface

NOTE insert WaterGroundSurfaceType, WaterGroundSurface UML

WaterClosureSurfaceType, WaterClosureSurface

NOTE insert WaterClosureSurfaceType, WaterClosureSurface UML

Transportation

The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and func-tional as well as on geometrical/topological aspects. Transportation features are represented as a linear network in LOD0. Starting from LOD1, all transportation features are geometrically described by 3D surfaces. The areal modelling of transportation features allows for the application of geometric route planning algorithms. This can be useful to determine restrictions and manoeuvres required along a transportation route. This information can also be employed for trajectory planning of mobile robots in the real world or the automatic placement of avatars (virtual people) or vehicle models in 3D visualisations and training simulators. The transportation model of CityGML is provided by the thematic extension module Transportation (cf. chapter 7).

The main class is TransportationComplex, which represents, for example, a road, a track, a railway, or a square. Fig. 57 illustrates the four different thematic classes.

NOTE insert Fig 57

A TransportationComplex is composed of the parts TrafficArea and AuxiliaryTrafficArea. Fig. 58 depicts an example for a LOD2 TransportationComplex configuration within a virtual 3D city model. The Road consists of several TrafficAreas for the sidewalks, road lanes, parking lots, and of AuxiliaryTrafficAreas below the raised flower beds.

NOTE insert Fig 58

Fig. 59 depicts the UML diagram of the transportation model, for the XML schema definition see annex A.10.

NOTE insert Fig 59

The road itself is represented as a TransportationComplex, which is further subdivided into TrafficAreas and AuxiliaryTrafficAreas. The TrafficAreas are those elements, which are important in terms of traffic usage, like car driving lanes, pedestrian zones and cycle lanes. The AuxiliaryTrafficAreas are describing further elements of the road, like kerbstones, middle lanes, and green areas.

TransportationComplex objects can be thematically differentiated using the subclasses Track, Road, Railway, and Square. Every TransportationComplex has the attributes class, function and usage whose possible values can be enumerated in code lists (chapter 10.7.4 and annex C.8). The attribute class describes the classification of the object, function describes the purpose of the object, for example national motorway, country road, or airport, while the attribute usage can be used, if the actual usage differs from the function. The attributes function and usage can occur multiple times.

In addition both TrafficArea and AuxiliaryTrafficArea may have the attributes class, function, usage, and sur-faceMaterial. The attribute class describes the classification of the object. For TrafficArea, function describes, if the object for example may be a car driving lane, a pedestrian zones, or a cycle lane, while the usage attribute indicates which modes of transportation can use it (e.g. pedestrian, car, tram, roller skates). The attribute sur-faceMaterial specifies the type of pavement and may also be used for AuxiliaryTrafficAreas (e.g. asphalt, con-crete, gravel, soil, rail, grass). The function attribute of the AuxiliaryTrafficArea defines, for example kerbstones, middle lanes, or green areas. The possible values can also be specified in code lists.

The shape of each traffic area is defined by a surface geometry. Additional metadata may be defined by using attributes from pre-defined catalogues. This affects the class, function and usage of each traffic area as well as its surface material. The attribute catalogues may be customer- or country-specific. The following tables show examples for various kinds of TrafficArea:

Table 4. Examples of TrafficArea

Example	Country Road	Motorway Entry
TransportationComplex – Function	road	road
TrafficArea – Usage	car, truck, bus, taxi, motorcycle	car, truck, bus, taxi, motorcycle
TrafficArea – Function	driving lane	motorway_entry
TrafficArea – SurfaceMaterial	asphalt	concrete

TransportationComplex is a subclass of _TransportationObject and of the root class _CityObject. The geomet-rical representation of the TransportationComplex varies through the different levels of detail. Since Transporta-tionComplex is a subclass of _CityObject and hence a feature, it inherits the attribute gml:name. The street name is also stored within the gml:name property of the Road feature.

In the coarsest LOD0 the transportation complexes are modelled by line objects establishing a linear network. On this abstract level, path finding algorithms or similar analyses can be executed. It also can be used to generate schematic drawings and visualisations of the transport network. Since this abstract definition of transportation network does not contain explicit descriptions of the transportation objects, it may be task of the viewer applica-tion to generate the graphical visualisation, for example by using a library with style-definitions (width, color resp. texture) for each transportation object.

Starting from LOD1 a TransportationComplex provides an explicit surface geometry, reflecting the actual shape of the object, not just its centerline. In LOD2 to LOD4, it is further subdivided thematically into TrafficAreas, which are used by transportation, such as cars, trains, public transport, airplanes, bicycles or pedestrians and in AuxiliaryTrafficAreas, which are of minor importance for transportation purposes, for example road markings, green spaces or flower tubs. The different representations of a TransportationComplex for each LOD are illus-trated in Fig. 60.

NOTE insert Fig60

In LOD0 areal transportation objects like squares should be modelled in the same way as in GDF, the ISO standard for transportation networks, which is used in most car navigation systems. In GDF a square is typically represented as a ring surrounding the place and to which the incident roads connect. CityGML does not cover further functional aspects of transportation network models (e.g. speed limits) as it is intended to complement and not replace existing standards like GDF. However, if specific functional aspects have to be associated with CityGML transportation objects, generic attributes provided by CityGML's Generics module (cf. chapter 10.12) can be used. Moreover, further objects of interest can be added from other information systems by the use of ExternalReferences (see chapter 6.11). For example, GDF datasets, which provide additional information for car navigation, can be used for simulation and visualisation of traffic flows. The values of the object attributes can be augmented or replaced using the concept of dictionaries (see chapter 6.6). These directories may be country- or user-specific (especially for country-specific road signs and signals).

NOTE insert fig61

The following example shows a complex urban crossing. The picture on the left is a screenshot of an editor application for a training simulator, which allows the definition of road networks consisting of transportation objects, external references, buildings and vegetation objects. On the right, the 3D representation of the defined crossing is shown including all referenced static and dynamic models.

NOTE insert fig62

Transporatation Complex

AbstractTransportationObjectType, _TransportationObject

NOTE insert AbstractTransportationObjectType, _TransportationObject UML

_TransportationObject represents the abstract superclass for transportation objects. Future extensions of the CityGML transportation model shall be modelled as subclasses of this class.

TransportationComplexType, TransportationComplex

NOTE insert TransportationComplexType, TransportationComplex UML

This type and element describe transportation complexes like roads or railways which may be aggregated from different thematic components (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As a subclass of _CityObject, TransportationComplex inherits all attributes and relations, in particular an id, names, external references, and generalisation relations. Furthermore, it represents the superclass for thematically distinct types of transportation complexes.

Subclasses of Transportation Complexes

TrackType, Track

NOTE insert TrackType, Track UML

A Track is a small path mainly used by pedestrians. It is a subclass of TransportationComplex and thus inherits all its attributes and relations.

RoadType, Road

NOTE insert RoadType, Road UML

Road is intended to be used to represent transportation features that are mainly used by vehicles like cars, for example streets, motorways, and country roads. It is a subclass of TransportationComplex and thus inherits all its attributes and relations.

RailwayType, Railway

NOTE insert RailwayType, Railway UML

Railway represents routes that are utilised by rail vehicles like trams or trains. It is a subclass of Transportation-Complex and thus inherits all its attributes and relations.

SquareType, Square

NOTE insert SquareType, Square UML

A Square is an open area commonly found in cities (e.g. a plaza, market square). It is a subclass of Transporta-tionComplex and thus inherits all its attributes and relations.

Subdivisions of Transportation Complexes

TrafficAreaType, TrafficArea

NOTE insert TrafficAreaType, TrafficArea UML

AuxiliaryTrafficAreaType, AuxiliaryTrafficArea

NOTE insert AuxiliaryTrafficAreaType, AuxiliaryTrafficArea UML

Chapter 7. Clause containing normative material

Paragraph

7.1. Requirement Class A or Requirement A Example

Paragraph – intro text for the requirement class.

Use the following table for Requirements Classes.

Requirements Class	
http://www.opengis.net/spec/ABCD/m.n/req/req-class-a	
Target type	Token
Dependency	http://www.example.org/req/blah
Dependency	urn:iso:ts:iso:19139:clause:6
Requirement 1	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-1 requirement description
Requirement 2	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-2 requirement description
Requirement 3	http://www.opengis.net/spec/ABCD/m.n/req/req-class-a/req-name-3 requirement description

7.1.1. Requirement 1

Paragraph - intro text for the requirement.

Use the following table for Requirements, number sequentially.

Requirement 1	/req/req-class-a/req-name-1 Requirement 'shall' statement

Dictionary tables for requirements can be added as necessary. Modify the following example as needed.

Names	**Definition**	**Data types and values**	**Multiplicity and use**
name 1	definition of name 1	float	One or more (mandatory)

Names	Definition	Data types and values	Multiplicity and use
name 2	definition of name 2	character string type, not empty	Zero or one (optional)
name 3	definition of name 3	GML:: Point PropertyType	One (mandatory)

Annex A: Conformance Class Abstract Test Suite (Normative)

NOTE Ensure that there is a conformance class for each requirements class and a test for each requirement (identified by requirement name and number)

A.1. Conformance Class A

A.1.1. Requirement 1

Test id:	/conf/conf-class-a/req-name-1
Requirement:	/req/req-class-a/req-name-1
Test purpose:	Verify that…
Test method:	Inspect…

A.1.2. Requirement 2

Annex B: Title ({Normative/Informative})

NOTE Place other Annex material in sequential annexes beginning with "B" and leave final two annexes for the Revision History and Bibliography

Annex C: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version

Annex D: Bibliography

NOTE

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

- For citations in the text please use square brackets and consecutive numbers: [1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, http://Website-Url

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015).