Open Geospatial Consortium Inc.

Date: 2006-07-28

Reference number of this document: OGC 06-057r1

Version: 0.3.0

Category: Candidate OpenGIS® Implementation Specification

Editors: Gerhard Gröger, Thomas H. Kolbe, Angela Czerwinski

Candidate OpenGIS® CityGML Implementation Specification
(City Geography Markup Language)

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

Developed by the Special Interest Group 3D (SIG 3D), 2006
www.citygml.org

Warning

This document is not an OGC Standard. It is distributed by the CAD-GIS Working Group for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation. Submittal process details and an open discussion forum are available on the OGC Network at this link: http://www.ogcnetwork.net/gml-citygml

Document type: Candidate OpenGIS® Implementation Specification
Document subtype: Application Schema for the Geography Markup Language 3 (GML 3.1.1)
Document stage: Discussion Paper
Document language: English
## Contents

1 Introduction ............................................................................................................... 6
   1.1 Motivation .......................................................................................................... 6
   1.2 Historical background ........................................................................................ 6

2 Scope ........................................................................................................................... 7

3 Conformance ............................................................................................................. 7

4 Normative references................................................................................................ 7

5 Conventions ............................................................................................................... 8
   5.1 Abbreviated terms .............................................................................................. 8
   5.2 UML Notation .................................................................................................... 9
   5.3 XML-Schema ..................................................................................................... 9

6 Overview of CityGML ............................................................................................ 10

7 General characteristics of CityGML ..................................................................... 11
   7.1 Multi-scale modelling (5 levels of detail, LOD) .............................................. 11
   7.2 Coherent semantical-geometrical modeling ..................................................... 12
   7.3 Closure surfaces ............................................................................................... 12
   7.4 Terrain Intersection Curve (TIC) ..................................................................... 13
   7.5 Dictionaries and code lists for enumerative attributes ..................................... 13
   7.6 External references ........................................................................................... 14
   7.7 City object groups ............................................................................................ 14
   7.8 Generic city objects and attributes ................................................................... 14
   7.9 Material and texture ......................................................................................... 14
   7.10 Prototypic objects / scene graph concepts ........................................................ 15

8 Geometry model ...................................................................................................... 16
   8.1 Geometric-topological model........................................................................... 16
   8.2 Surfaces with materials .................................................................................... 19
   8.3 Implicit geometries, prototypic objects, scene graph concepts........................ 21

9 Thematic model ....................................................................................................... 24
   9.1 Top level classes ............................................................................................... 24
   9.2 Digital Terrain Model (DTM) .......................................................................... 27
   9.3 Building model .................................................................................................. 31
      9.3.1 Building and building part............................................................................ 33
      9.3.2 Boundary surfaces ....................................................................................... 37
      9.3.3 BuildingInstallations.................................................................................... 40
      9.3.4 Openings ..................................................................................................... 40
      9.3.5 Building Interior ........................................................................................ 41
CityGML has been developed by the members of the Special Interest Group 3D of the initiative Geodata Infrastructure North-Rhine Westphalia (GDI NRW) so far. The present CityGML version is being addressed as CityGML 1.0 within the SIG 3D.


The preparation of the English document version and the European discussion has been supported by the European Spatial Data Research Organization (EuroSDR; formerly known as OEEPE) in a current EuroSDR Commission III project.

For further information see [http://www.eurosdr.net](http://www.eurosdr.net)

### ii. Submitting organizations

- Institute for Cartography and Geoinformation, University of Bonn
- Special Interest Group 3D (SIG 3D) of the Geodata Infrastructure North-Rhine Westphalia (GDI NRW).

### iii. Participants in development

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Thomas H. Kolbe,</td>
<td>Institute for Cartography and Geoinformation, University of Bonn</td>
<td><a href="mailto:kolbe@ikg.uni-bonn.de">kolbe@ikg.uni-bonn.de</a></td>
</tr>
<tr>
<td>Dr. Gerhard Gröger</td>
<td></td>
<td><a href="mailto:groeger@ikg.uni-bonn.de">groeger@ikg.uni-bonn.de</a></td>
</tr>
<tr>
<td>Prof. Dr. Lutz Plümer</td>
<td></td>
<td><a href="mailto:pluemer@ikg.uni-bonn.de">pluemer@ikg.uni-bonn.de</a></td>
</tr>
<tr>
<td>Angela Czerwinski</td>
<td></td>
<td><a href="mailto:czerwinski@ikg.uni-bonn.de">czerwinski@ikg.uni-bonn.de</a></td>
</tr>
<tr>
<td>Dirk Dörschlag</td>
<td></td>
<td><a href="mailto:doerschlag@ikg.uni-bonn.de">doerschlag@ikg.uni-bonn.de</a></td>
</tr>
<tr>
<td>Ulrich Gruber,</td>
<td>District Administration Recklinghausen, Cadastre Department</td>
<td><a href="mailto:ulrich.gruber@kreis-recklinghausen.de">ulrich.gruber@kreis-recklinghausen.de</a></td>
</tr>
<tr>
<td>Birgit Joemann</td>
<td></td>
<td><a href="mailto:birgit.joemann2@kreis-re.de">birgit.joemann2@kreis-re.de</a></td>
</tr>
<tr>
<td>Dr. Joachim Benner</td>
<td>Institute for Applied Computer Science, Helmholtz Research Center Karlsruhe</td>
<td></td>
</tr>
<tr>
<td>Dr. Klaus Leinemann</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frank Bildstein</td>
<td>Rheinmetall Defence Electronics</td>
<td><a href="mailto:bildstein.f@rheinmetall-de.com">bildstein.f@rheinmetall-de.com</a></td>
</tr>
<tr>
<td>Rüdiger Drees</td>
<td>T-Mobile</td>
<td><a href="mailto:ruediger.drees@t-mobile.de">ruediger.drees@t-mobile.de</a></td>
</tr>
<tr>
<td>Andreas Kohlhaas</td>
<td>GItStec GmbH</td>
<td><a href="mailto:andreas.kohlhaas@GItStec-online.de">andreas.kohlhaas@GItStec-online.de</a></td>
</tr>
<tr>
<td>Frank Thiemann</td>
<td>Institute for Cartography and Geoinformatics, University of Hannover</td>
<td><a href="mailto:Frank.Thiemann@ikg.uni-hannover.de">Frank.Thiemann@ikg.uni-hannover.de</a></td>
</tr>
<tr>
<td>Martin Degen</td>
<td>City of Dortmund Cadastre Department</td>
<td><a href="mailto:mdegen@stadtdo.de">mdegen@stadtdo.de</a></td>
</tr>
<tr>
<td>Prof. Dr. Jürgen Döllner</td>
<td>Hasso Plattner Institute for Software Technology, University of Potsdam</td>
<td><a href="mailto:juergen.doellner@hpi.uni-potsdam.de">juergen.doellner@hpi.uni-potsdam.de</a></td>
</tr>
<tr>
<td>Heinrich Geerling</td>
<td>Architekturbüro Geerling</td>
<td><a href="mailto:heinrich@geerling.de">heinrich@geerling.de</a></td>
</tr>
<tr>
<td>Dr. Frank Knospe</td>
<td>City of Essen Cadastre and Mapping Department</td>
<td><a href="mailto:frank.knospe@amt62.essen.de">frank.knospe@amt62.essen.de</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="mailto:fknospe@gmx.de">fknospe@gmx.de</a></td>
</tr>
<tr>
<td>Hardo Müller</td>
<td>Snowflake Software Ltd.</td>
<td><a href="mailto:Hardo.Mueller@snowflakesoftware.co.uk">Hardo.Mueller@snowflakesoftware.co.uk</a></td>
</tr>
<tr>
<td>Martin Rechner</td>
<td>rechner logistik</td>
<td><a href="mailto:mail@rec-log.de">mail@rec-log.de</a></td>
</tr>
<tr>
<td>Jörg Haist</td>
<td>Fraunhofer Institute for Computer Graphics (IGD)</td>
<td><a href="mailto:joerg.haist@igd.fraunhofer.de">joerg.haist@igd.fraunhofer.de</a></td>
</tr>
<tr>
<td>Daniel Holweg</td>
<td></td>
<td><a href="mailto:daniel.holweg@igd.fraunhofer.de">daniel.holweg@igd.fraunhofer.de</a></td>
</tr>
<tr>
<td>Prof. Dr. Peter A. Henning</td>
<td>Faculty for Computer Science University of Applied Sciences, Karlsruhe</td>
<td><a href="mailto:p.henning@fh-karlsruhe.de">p.henning@fh-karlsruhe.de</a></td>
</tr>
<tr>
<td>Carsten Rünsdorf, Dave Capstick, Marc Pendlington</td>
<td>British Ordnance Survey</td>
<td><a href="mailto:Carsten.Roensdorf@ordnancesurvey.co.uk">Carsten.Roensdorf@ordnancesurvey.co.uk</a></td>
</tr>
<tr>
<td>Rolf Wegener</td>
<td>State Cadastre and Mapping Agency of North-Rhine Westphalia</td>
<td><a href="mailto:wegener@lverma.nrw.de">wegener@lverma.nrw.de</a></td>
</tr>
</tbody>
</table>

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
### Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Editor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.06</td>
<td>0.1.0</td>
<td>Czerwinski, Kolbe, Gröger</td>
<td>Initialisation of the document</td>
</tr>
<tr>
<td>22.2.06</td>
<td>0.1.0</td>
<td>Gröger, Gruber</td>
<td>Additions to UML diagrams</td>
</tr>
<tr>
<td>26.4.06</td>
<td>0.2.0</td>
<td>Kolbe, Gröger, Czerwinski</td>
<td>Release of CityGML draft specification to EuroSDR</td>
</tr>
<tr>
<td>3.6.06</td>
<td>0.3.0</td>
<td>Kolbe, Gröger, Czerwinski</td>
<td>Changes on property names, some attributes added. Release of CityGML specification document to OGC</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

An increasing number of cities and companies are building virtual 3D city models for different application areas like urban planning, mobile telecommunication, disaster management, 3D cadastre, tourism, vehicle and pedestrian navigation, facility management, or environmental simulations. Furthermore, in the implementation of the European Environmental Noise Directive (END, 2002/49/EC) 3D geoinformation and 3D city models play an important role.

In recent years, most virtual 3D city models have been defined as purely graphical or geometrical models, neglecting the semantic and topological aspects. Thus, these models could almost only be used for visualization purposes but not for thematic queries, analysis tasks, or spatial data mining. Since the limited reusability of models inhibits the broader use of 3D city models, a more general modelling approach has to be taken in order to satisfy the information needs of the various application fields.

CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the possibility of selling the same data to customers from different application fields. The targeted application areas explicitly include city planning, architectural design, touristic and leisure activities, environmental simulation, mobile telecommunication, disaster management, homeland security, real estate management, vehicle and pedestrian navigation, and training simulators.

CityGML is designed as an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is implemented as an application schema of the Geography Markup Language 3 (GML3), the extensible international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium (OGC) and the ISO TC211. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS.

CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical and appearance properties. “City” is broadly defined to comprise not just built structures, but also elevation, vegetation, water bodies, “city furniture”, and more. Included are generalization hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is applicable for large areas and small regions and can represent the terrain and 3D objects in different levels of detail simultaneously. Since from simple, single scale models without topology and few semantics up to very complex multi-scale models with full topology and fine-grained semantical differentiations can be represented, CityGML enables lossless information exchange between different GI systems and users.

1.2 Historical background

Since 2002, CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW) in Germany. The SIG 3D is an open group consisting of more than 70 companies, municipalities, and research institutions from Germany, Great Britain, Switzerland, and Austria working on the development and commercial exploitation of interoperable 3D models and geovisualization. Another result of the work from the SIG 3D is the proposition of the Web 3D Service (W3DS), a 3D portrayal service that is also being discussed in the Open Geospatial Consortium (OGC Doc.No. 05-019).

A subset of CityGML has been successfully implemented and evaluated in the project “Pilot 3D” of the GDI NRW in 2005. Participants came from all over Germany and demonstrated city planning scenarios and touristic applications. The new official 3D city model of Berlin is based on the CityGML data model and employs CityGML as the exchange format between database, editor, and presentation systems.

By the beginning of 2006, a CityGML project within EuroSDR (European Spatial Data Research) has been started aiming at the European harmonization of 3D city modeling.
2 Scope

This document is a candidate OpenGIS specification for the representation, storage and exchange of virtual 3D city and regional models.

3 Conformance

XML files must be validated against the CityGML schema and fulfill all further requirements of the CityGML specification. All objects modeled in CityGML shall be modeled accordingly in any application.

4 Normative references

The following referenced documents are indispensable for the application of the CityGML specification. The geometry model of GML 3.1.1 is used except for some added concepts like implicit geometries (see chapter 8.3). Furthermore, a concept on positioning textures on surfaces is added (see chapter 8.2), which is adopted from the 3D computer graphics standard X3D (Web 3D 2004), the successor of VRML97.

ISO 8601:2000, Data elements and interchange formats – Information interchange Representation of dates and times
ISO 19105:2000, Geographic information – Conformance and testing
ISO 19107:2003, Geographic Information – Spatial Schema
ISO 19109:2005, Geographic Information – Rules for Application Schemas
ISO 19111:2003, Geographic information – Spatial referencing by coordinates
ISO 19115:2003, Geographic Information – Metadata
ISO 19123:2005, Geographic Information – Coverages
ISO 19136:–, Geographic Information – Geography Markup Language (GML 3.1.1) (to be published)
ISO/TS 19139:–, Geographic Information – Metadata – Implementation Specification (to be published)
OpenGIS® Abstract Specification Topic 0, Overview, OGC document 99-100r1
OpenGIS® Abstract Specification Topic 8, Relations between Features, OGC document 99-108r2
IETF RFC 2045 & 2046, Multipurpose Internet Mail Extensions (MIME). (November 1996)
W3C XLink, XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)
W3C XMLName, Namespaces in XML. W3C Recommendation (14 January 1999)
W3C XML Base, XML Base, W3C Recommendation (27 June 2001)
W3C XML, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)
OASIS (Organization for the Advancement of Structured Information Standards): extensible Address Language (xAL v2.0).

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
5 Conventions

5.1 Abbreviated terms

The following abbreviated terms are used in this document:

- AEC: Architecture, Engineering, Construction
- ALKIS: German National Standard for Cadastral Information
- ATKIS: German National Standard for Topographic and Cartographic Information
- B-Rep: Boundary Representation
- CAD: Computer Aided Design
- CAAD: Computer Aided Architectural Design
- CSG: Constructive Solid Geometry
- DTM: Digital Terrain Model
- DXF: Drawing Exchange Format
- EuroSDR: European Spatial Data Research Organisation
- FM: Facility Management
- GDF: Geographic Data Files
- GDI NRW: Geodata Infrastructure North-Rhine Westphalia
- GML: Geography Markup Language
- IAI: International Alliance for Interoperability
- IETF: Internet Engineering Task Force
- IFC: Industry Foundation Classes
- ISO: International Organization for Standardization
- LOD: Level of Detail
- OASIS: Organization for the Advancement of Structured Information Standards
- OGC: Open Geospatial Consortium
- SIG 3D: Special Interest Group 3D
- TC211: ISO Technical Committee 211
- TIC: Terrain Intersection Curve
- TIN: Triangulated Irregular Network
- UML: Unified Modeling Language
- URI: Uniform Resource Identifier
- VRML: Virtual Reality Modeling Language
- W3C: World Wide Web Consortium
- W3DS: OGC Web 3D Service
- WFS: OGC Web Feature Service
- X3D: Open Standards XML-enabled 3D file format of the Web 3D Consortium
- XML: Extensible Markup Language
- xAL: OASIS extensible Address Language
5.2 UML Notation

The CityGML specification is presented in this document in diagrams using the Unified Modeling Language (UML) static structure diagram (see Booch et al. 1997). The UML notations used in this standard are described in the diagram below (Fig. 1).

A special symbol is introduced to denote the associations between a feature and its geometry in different levels of detail (LoD), c.f. Fig. 2. The dashed line means that the association occurs for each LoD in the interval [a..b]. This symbol extends the standard UML notation.

The following stereotypes are used:
<<Feature>> represents the geometry of an object
<<Feature>> represents a thematic feature
<<ExternalCodeList>> enumerates the valid attribute values (see chapter 7.5)
<<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.

5.3 XML-Schema

The normative parts of the specification use the W3C XML schema language to describe the grammar of conformant CityGML data instances. XML schema is a rich language with many capabilities. While a reader who is unfamiliar with an XML schema may be able to follow the description in a general fashion, this specification is not intended to serve as an introduction to XML schema. In order to have a full understanding of this candidate specification, it is necessary for the reader to have a reasonable knowledge of XML schema.
6 Overview of CityGML

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 3 (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 to reuse the same in different application fields.

CityGML not only represents the graphical appearance of city models but specifically addresses the representation of the semantic and thematic properties, taxonomies and aggregations. CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models (chapter 8). The base class of all objects is CityObject. All objects inherit the properties from CityObject.

The thematic model of CityGML employs the geometry model for different thematic fields like Digital Terrain Models, sites (i.e. buildings; future extensions of CityGML will also include explicit models for bridges and tunnels), vegetation (solitary objects and also area and volumetric biotopes), water bodies, transportation facilities, and city furniture (chapter 9). Further objects, which are not explicitly modelled yet, can be represented using the concept of generic objects and attributes (chapter 7.8). Spatial objects of equal shape which appear many times at different positions like e.g. trees, can also be modelled as prototypes and used multiple times in the city model (chapter 8.3). A grouping concept allows the combination of single 3D objects, e.g. buildings to a building complex (chapter 7.7). Objects which are not geometrically modelled by closed solids must be virtually sealed in order to compute their volume (e.g. pedestrian underpasses, tunnels, or airplane hangars). They can be sealed using ClosureSurfaces (chapter 7.3). The concept of the TerrainIntersectionCurve is introduced to integrate 3D objects with the Digital Terrain Model at their correct positions in order to prevent e.g. buildings from floating over or sinking into the terrain (chapter 7.4).

CityGML differentiates five consecutive Levels Of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (chapter 7.1). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Generalization relations allow the explicit representation of aggregated objects over different scales. Object surfaces can be assigned textures and material properties like e.g. colors (chapter 8.2).

Furthermore, objects can have external references to corresponding objects in external datasets (chapter 7.6). Enumerative object attributes are restricted to external code lists and values defined in external, redefinable dictionaries (chapter 7.5).
7 General characteristics of CityGML

7.1 Multi-scale modelling (5 levels of detail, LOD)

CityGML supports different Levels of Detail (LOD). LODs are required to reflect independent data collection processes with differing application requirements. Further, LODs facilitate efficient visualization and data analysis (see Fig. 3). In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualization of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated.

The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model, over which an aerial image or a map may be draped. LOD1 is the well-known blocks model comprising prismatic buildings with flat roofs. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated surfaces. Vegetation objects may also be represented. LOD3 denotes architectural models with detailed wall and roof structures, balconies, bays and projections. High-resolution textures can be mapped onto these structures. In addition, detailed vegetation and transportation objects are components of a LOD3 model. LOD4 completes a LOD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture.

 LOD0  LOD1  LOD2

 LOD3  LOD4

Fig. 3: The five levels of detail (LOD) defined by CityGML (source: IKG Uni Bonn)

LODs are also characterized by differing accuracies and minimal dimensions of objects (Tab. 1). The accuracy requirements given in this candidate specification are debatable and should be considered as discussion proposals. Accuracy is described as standard deviation \( \sigma \) of the absolute 3D point coordinates. Relative 3D point accuracy will be added in a future version of CityGML as it is typically much higher than the absolute accuracy. In LOD1, the positional and height accuracy of points must be 5m or less, while all objects with a footprint of at least 6m by 6m have to be considered. The positional and height accuracy of LOD2 must be 2m or better. In this LOD, all objects with a footprint of at least 4m \( \times \) 4m have to be considered. Both types of accuracies in LOD3 are 0.5m, and the minimal footprint is 2m \( \times \) 2m. Finally, the positional and height accuracy of LOD4 must be 0.2m or less. By means of these figures, the classification in five LOD may be used to assess the quality of 3D city model datasets. The LOD categorization makes datasets comparable and provides support for their integration.

<table>
<thead>
<tr>
<th>Model scale description</th>
<th>LOD0</th>
<th>LOD1</th>
<th>LOD2</th>
<th>LOD3</th>
<th>LOD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes of accuracy</td>
<td>lowest</td>
<td>low</td>
<td>middle</td>
<td>high</td>
<td>very high</td>
</tr>
<tr>
<td>Absolut 3D point accuracy (position / height)</td>
<td>lower than LOD1</td>
<td>5/5m</td>
<td>2/2m</td>
<td>0.5/0.5m</td>
<td>0.2/0.2m</td>
</tr>
<tr>
<td>Generalisation</td>
<td>maximal</td>
<td>object blocks as</td>
<td>objects as</td>
<td>object as real</td>
<td>constructive</td>
</tr>
</tbody>
</table>

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
Whereas in CityGML each object can have a different representation for every LOD, often different objects from the same LOD will be generalized to be represented by an aggregate object in a lower LOD. CityGML supports the aggregation / decomposition by providing an explicit generalisation association between any CityObjects (further details see UML diagram in chapter 9.10).

7.2 **Coherent semantical-geometrical modeling**

One of the most important design principles for CityGML is the coherent modeling of semantics and geometrical/topological properties. At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features. Thus the part-of-relationship between features can be derived at the semantic level only, without considering geometry. However, at the spatial level, geometry objects are assigned to features representing their spatial location and extent. So the model consists of two hierarchies: the semantic and the geometrical in which the corresponding objects are linked by relationships. The advantage of this approach is that it can be navigated in both hierarchies and between both hierarchies arbitrarily, for answering thematic and/or geometrical queries or performing analyses.

If both hierarchies exist for a specific object, they must be coherent (i.e. it must be assured that they match and fit together). For example, if a wall of a building has two windows and a door on the semantic level, then the geometry representing the wall must contain also the geometry parts of both windows and the door.

7.3 **Closure surfaces**

Objects, which are not modelled by a volumetric geometry, must be virtually closed in order to compute their volume (e.g. pedestrian underpasses or airplane hangars). They can be sealed using ClosureSurfaces. ClosureSurfaces are special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualizations.

The concept of ClosureSurfaces is also employed to model the entrances of subsurface objects. Those objects like tunnels or pedestrian underpasses have to be modelled as closed solids in order to compute their volume, for example in flood simulations. The entrances to subsurface objects also have to be sealed to avoid holes in the digital terrain model (see Fig. 4). However, in close-range visualizations the entrance must be treated as open. Thus, ClosureSurfaces are an adequate way to model those entrances.
Fig. 4: Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual ClosureSurface, which is both part of the DTM and the subsurface object (right) (graphic: IKG Uni Bonn).

7.4 Terrain Intersection Curve (TIC)

A crucial issue in city modelling is the integration of buildings and the terrain. Problems arise if buildings float over or sink into the terrain. This is particularly the case if terrains and buildings in different LOD are combined, or if they come from different providers (Gröger and Kolbe 2003). To overcome this problem, the TerrainIntersectionCurve (TIC) of a building is introduced. This curve denotes the exact position, where the terrain touches the building, and is represented by one or more closed rings surrounding the building (see Fig. 5). If the building has a courtyard, the TIC consists of two closed rings: one ring representing the courtyard boundary, and one which describes the building’s outer boundary. This information can be used to integrate the building and a terrain by ‘pulling up’ or ‘pulling down’ the surrounding terrain to fit the TerrainIntersectionCurve. The DTM may be locally warped to fit the TIC. By this means the TIC also ensures the correct positioning of textures or the matching of object textures with the DTM. Since the intersection with the terrain may differ depending on the LOD, a building may have different TerrainIntersectionCurves for all LOD.

Fig. 5: TerrainIntersectionCurve for a building (left, black) and a tunnel object (right, white). The tunnel’s hollow space is sealed by a triangulated ClosureSurface (graphic: IKG Uni Bonn).

7.5 Dictionaries and code lists for enumerative attributes

Attributes, which are used to classify objects, often have values that are restricted to a number of discrete values. An example is the attribute roof type, whose attribute values typically are saddle back roof, hip roof, semi-hip roof, flat roof, pent roof, or tent roof. If such an attribute is typed as string, misspellings or different names for the same notion obstruct interoperability. In CityGML such classifying of attributes are specified as CodeLists and implemented by GML3 Dictionaries (c.f. Cox et al. 2004). Such a structure enumerates all possible values of the attribute in an external file, assuring that the same name is used for the same notion. In addition, the translation of attribute values into other languages is facilitated. Dictionaries and code lists may be extended or redefined by users. They can have references to existing models.
7.6 **External references**

3D objects are often derived from or have relations to objects in other databases or data sets. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set, or may be derived from an architectural model (Fig. 6). The reference of a 3D object to its corresponding object in an external data set is essential, if an update must be propagated or if additional data is required, for example the name and address of a building’s owner in a cadastral information system or information on antennas and doors in a facility management system. In order to supply such information, each CityObject may have **External References** to corresponding objects in external data sets (for the UML diagram see Fig. 14; and for XML schema definition see chapter 10.1.5). Such a reference denotes the external information system and the unique identifier of the object in this system. Both are specified as a **Uniform Resource Identifier (URI)**, which is a generic format for references to any kind of resources in the internet. The generic concept of external references permits for any CityObject an arbitrary number of links to corresponding objects in external information systems (e.g. ALKIS, ATKIS, Ordnance Survey Mastermap, GDF, etc.).

![Fig. 6: External references (graphic: IKG Uni Bonn).](image)

7.7 **City object groups**

The grouping concept of CityGML allows the aggregation of arbitrary city objects according to user-defined criteria, and to represent and transfer these aggregations as part of a city model (for the UML diagram see chapter 9.9; XML schema definition see chapter 10.1.4). A group may be assigned one or more names and may be further classified by specific attributes, for example, "escape route from room no. 43 in house no. 1212 in a fire scenario" as a name and "escape route" as type. Each member of the group can optionally be assigned a role name, which specifies the role this particular member plays in the group. This role name may e.g. describe the sequence number of this object in an escape route, or in the case of a building complex, denote the main building.

A group may contain other groups as members, allowing nested grouping of arbitrary depth.

7.8 **Generic city objects and attributes**

In practical applications the objects within specific 3D city models will most likely contain attributes which are not explicitly modelled in CityGML. Moreover, there might be 3D objects which are not covered by the thematic classes of CityGML. In order to support also the exchange of such data, the concept of generic objects (**GenericCityObject**) and attributes (**CityObjectGenericAttribute**) has been added to CityGML. The corresponding UML diagram is given in chapter 9.10 and the XML schema definition in chapter 10.1.3.

The current version of CityGML does not include explicit thematic models for bridges, tunnels, and walls. They will be added in a future version. In the meantime, these objects should be stored or exchanged using generic objects and attributes.

7.9 **Material and texture**

Information about surface materials and textures are considered as an integral part of virtual 3D city models. Since GML3 has no built-in concept for the representation of surface materials, CityGML extends the GML3 geometry model by the new class **TexturedSurface**, which allows to assign appearance properties (colors, shini-
ness, transparency) and textures to any 3D surface (for further description see chapter 8.2). The definition of the appearance properties is adopted from the X3D specification.

7.10 Prototypic objects / scene graph concepts

In CityGML objects of equal shape like trees and other vegetation objects, traffic lights, or traffic signs can be represented as prototypes which are instantiated multiple times at different locations (Fig. 7). The geometry of prototypes is defined in local coordinate systems. Every instance is represented by a reference to the prototype, a base point in the world coordinate reference system, and a transformation matrix that facilitates scaling, rotation, and translation of the prototype. The principle is adopted from the concept of scene graphs used in computer graphics standards like VRML and X3D. As the GML3 geometry model does not provide support for scene graph concepts, it is implemented as an extension to the GML3 geometry model (further description see chapter 8.3).

Fig. 7: Examples of prototypic shapes (source: Rheinmetall Defence Electronics).
8 Geometry model

Spatial properties of CityGML features are represented by objects of GML3’s geometry model. This model is based on the standard ISO 19107 ‘Spatial Schema’ (Herring 2001), representing 3D geometry according to the well-known Boundary Representation (B-Rep, cf. Foley et al. 1995). CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. This subset is depicted in Fig. 8 and Fig. 9. On the other hand, CityGML extends the GML3 geometry model by adding concepts for representing textures and other material properties of surfaces (chapter 8.2). Furthermore, GML3’s explicit Boundary Representation is extended by scene graph concepts, which allow the representation of the geometry of features with the same shape implicitly and thus more space efficiently (chapter 8.3).

8.1 Geometric-topological model

The geometry model of GML 3 consists of primitives, which may be combined to form complexes, composite geometries or aggregates. For each dimension, there is a geometrical primitive: a zero-dimensional object is a Point, a one-dimensional a _Curve, a two-dimensional a Surface, and a three-dimensional a Solid (Fig. 8). Each geometry can have its own coordinate reference system. A solid is bounded by surfaces and a surface by curves. In CityGML 1.0, a curve is restricted to be a straight line, thus only the GML3 class LineString is used. Surfaces in CityGML are represented by Polygons, which define a planar geometry, i.e. the boundary and all interior points are required to be located in one single plane.

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in Fig. 10). In an Aggregate, the spatial relationship between components is not restricted. They may be disjoint, overlapping, touching, or disconnected. GML3 provides a special aggregate for each dimension, a MultiPoint, a MultiCurve, a MultiSurface or a MultiSolid (see Fig. 9). In contrast to aggregates, a Complex is topologically structured: its parts must be disjoint, must not overlap and are allowed to touch, at most, at their boundaries or share parts of their boundaries. A Composite is a special complex provided by GML3. It can only contain elements of the same dimension. Its elements must be disjoint as well, but they must be topologically connected along their boundaries. A Composite can be a CompositeSolid, a CompositeSurface, or CompositeCurve. (c.f. Fig. 8).
An **OrientableSurface** is a surface with an explicit orientation, i.e. two sides, front and back, can be distinguished. This may be used to assign textures to specific sides of a surface, or to distinguish the exterior and the interior side of a surface when bounding a solid. Please note, that Curves and Surfaces have a default orientation in GML which results from the order of the defining points. Thus, an OrientableSurface only has to be used, if the orientation of a given GML geometry has to be reversed.

**TriangulatedSurfaces** are special surfaces, which specify triangulated irregular networks often used to represent the terrain. While a **TriangulatedSurface** is a composition of explicit **Triangles**, the subclass **TIN** is used to represent a triangulation in an implicit way by a set of control points, defining the nodes of the triangles. The triangulation may be reconstructed using standard triangulation methods (Delaunay triangulation). In addition, break lines and stop lines define characteristic contour lines of the terrain.

The GML3 composite model realizes a **recursive aggregation** schema for every primitive type of the corresponding dimension. This aggregation schema allows the definition of nested aggregations (hierarchy of components). For example, a building geometry (**CompositeSolid**) can be composed of the house geometry (**CompositeSolid**) and the garage geometry (**Solid**), while the house’s geometry is further decomposed into the roof geometry (**Solid**) and the geometry of the house body (**Solid**).

CityGML provides the explicit modeling of topology, e.g. the sharing of geometry objects between features or other geometries. One part of space is represented only once by a geometry object, and is referenced by all features or more complex geometries which are defined or bounded by this geometry object. Thus redundancy is avoided and explicit topological relations between parts are maintained. Basically, there are three cases: First, two features may be defined spatially by the same geometry. For example, if a path is both a transportation feature and a vegetation feature, the surface geometry defining the path is referenced both by the transportation object and by the vegetation object. Second, geometry may be shared between a feature and another geometry. A geometry defining a wall of a building may be referenced twice: by the solid geometry defining the geometry of the building, and by the wall feature. Third, two geometries may reference the same geometry, which is in the
boundary of both. For example, a building and an adjacent garage may be represented by two solids. The surface
describing the area where both solids touch may be represented only once and it is referenced by both solids. As
it can be seen from Fig. 11, this requires partitioning of the respective surfaces. In general, Boundary
Representation only considers visible surfaces. However, to make topological adjacency explicit and to allow the
possibility of deletion of one part of a composed object without leaving holes in the remaining aggregate,
touching elements are included. Whereas touching is allowed, permeation of objects is not in order to avoid the
multiple representation of the same space. However, the use of topology in CityGML is optional.

In order to implement topology, CityGML uses the XML concept of *XLinks* provided by GML. Each geometry
object that should be shared by different geometric aggregates or different thematic features is assigned an
unique identifier, which may be referenced by a GML geometry property using a *href* attribute. CityGML does
not deploy the built-in topology package of GML3, which provides separate topology objects accompanying the
geometry. This kind of topology is very complex and elaborate. Nevertheless, it lacks flexibility when data sets,
which might include or neglect topology, should be covered by the same data model. The XLink topology is
simple and flexible, and nearly as powerful as the explicit GML3 topology model. However, a disadvantage of
the XLink topology is that navigation between topologically connected objects can only be performed in one
direction (from an aggregate to its components), not (immediately) bidirectional as it is the case for GML’s built-
in topology. An example for CityGML’s topology representation is given in the dataset listed in chapter 11.4.1.

The following excerpt of a CityGML example file defines a *gml:Polygon* with an id *wallSurface4711*, which is
part of the geometry property *lod2Surface* of a building. Another building being adjacent to the first building
references this polygon in its geometry property.

```xml
<building>
    ……
    <lod2Solid>
        ……
        <gml:surfaceMember>
            <gml:Polygon gml:id="wallSurface4711">…
                <gml:exterior>
                    <gml:LinearRing>
                        <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
                    </gml:LinearRing>
                </gml:exterior>
            </gml:Polygon>
            ……
        </gml:surfaceMember>
    </lod2Solid>
    ……
</building>
```

Fig. 11: Recursive aggregation of objects and geometries in CityGML (graphic: IKG Uni Bonn).
8.2 **Surfaces with materials**

Each surface or composite surface can be specialized to a *TexturedSurface*, which can be assigned *Materials* (colors, shininess, transparency) or *SimpleTextures* (Fig. 8 depicts the UML diagram, for XML schema definition see chapter 10.2.1).

The concept of positioning textures on surfaces complies with the 3D computer graphics standard X3D (web 3D 2004), a successor of VRML97. Because there has been no appropriate texturing concept in ISO 19107 and in GML3, CityGML adds the class *TexturedSurface* to the geometry model of GML 3.

A texture is specified as a raster image referenced by an *URI* (Uniform Resource Identifier), and can be an arbitrary resource, even in the internet. Textures are positioned by employing the concept of *texture coordinates*, i.e. each texture coordinate matches with exactly one 3D coordinate of the *TexturedSurface* (Fig. 12). The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry.

The color of a surface is defined by RGB values. These have to be in the range of 0 to 1. The *frontOpacity* and the *backOpacity* define the level of transparency of each surface. Their values have to be in the range of 0 to 1 as well, where 1 means completely covering and 0 denotes a completely transparent surface. The colors can be differentiated in *diffuseColor* (color when illuminated by a source of light), *emissiveColor* (color when self-illuminating) and *specularColor/shininess* (color when shiny surfaces).

Textures can be qualified by the attribute *textureType*. The *textureType* differentiates between textures, which are specific for a certain object (*specific*), and prototypic textures being typical for that object surface (*typical*). Textures may also be classified as *unknown*.

*Appearance* is derived from gml:AbstractGMLType to be referenced in an *appearance* property. The attribute gml:id is inherited, whose value may be referenced by a XLink. *Appearance* is the upper class of *Material* and *SimpleTexture*.

**TexturedSurfaceType, TexturedSurface**

![Fig. 12: Positioning and georeferencing of textures in CityGML (graphic: IKG Uni Bonn).](image-url)
TexturedSurface may have one or more appearance properties, which can either be a Material (Color,...) or a Texture. The _Appearance Element can either be represented inlined as an element of this type or by an XLink reference to a remote _Appearance element. Either the reference or the contained element must be given, but neither both nor none. The side of the surface the _Appearance refers to is given by the orientation attribute of the appearance property element, which refers to the corresponding sign attribute of the orientable surface: + means the side with positive orientation, and - the side with negative orientation.

_AppearanceType, _Appearance

MaterialType, Material

SimpleTextureType, SimpleTexture
### 8.3 Implicit geometries, prototypic objects, scene graph concepts

The concept of implicit geometries is an enhancement of the geometry model of GML3. It is used in CityGML’s vegetation model, for city furniture, and generic objects (see chapters 9.6, 9.7 and 9.10). The UML diagram is depicted in Fig. 13, the corresponding XML schema definition is provided in chapter 10.2.2.

An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, e.g. a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. This reference point also defines the CRS to which the world coordinates belong after the application of the transformation. In order to determine the absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way even a projection might be modeled by the transformation matrix.

The reason for using the concept of implicit geometries in CityGML is space efficiency. Since the shape of e.g. trees of the same species can be treated as identical, it would be inefficient to model the detailed geometry of each of the large number of trees explicitly. The concept of implicit geometries is similar to the well known concept of primitive instancing used for the representation of scene graphs in the field of computer graphics (Foley et al. 1995).

The term Implicit geometry refers to the principle that a geometry object with a complex shape can be simply represented by a base point and a transformation, implicitly unfolding the object’s shape at a specific location in the world coordinate system.

The shape of an ImplicitGeometry can be represented in an external file with a proprietary format, e.g. a VRML file, a DXF file, or a 3D Studio MAX file. The reference to the implicit geometry can be specified by an URI pointing to a local or remote file, or even to an appropriate web service. Alternatively, the shape can be defined by a GML3 geometry object. This has the advantage that it can be stored or exchanged inline within the CityGML dataset. Typically, the shape of the geometry is defined in a local coordinate system where the origin lies within or near to the object’s extent. If the shape is referenced by an URI, also the MIME type of the denoted object has to be specified (e.g. “model/vrml” for VRML models or “model/3d+xml” for X3D models).

The implicit representation of 3D object geometry has some advantages compared to the explicit modeling, which represents the objects using absolute world coordinates. It is more space-efficient, and thus more extensive scenes can be stored or handled by a system. The visualization is accelerated since 3D graphics cards support the scene graph concept. Furthermore, the usage of different shape versions of objects is facilitated, e.g. different seasons, since only the library objects have to be exchanged (see example in Fig. 34 on page 56).

---

**Fig. 13: UML diagram of ImplicitGeometries.**
ImplicitGeometryType

An example for an implicit geometry is given by the following city furniture object (c.f. section 9.7), which is represented by a geometry in LOD 2:

```xml
<CityFurniture>
  <class>1080</class> <!-- “traffic light”; declared in external code list (CityFurnitureClassType) in annex A -->
  <function>1000</function> <!-- “traffic”; declared in external code list (CityFurnitureFunctionType) in annex A -->
  <lod2ImplicitRepresentation>
    <ImplicitGeometry>
      <mimeType>model/vrml</mimeType>
      <libraryObject>
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </libraryObject>
      <referencePoint>
        <gml:Point srsName="EPSG:31467"> <!-- Gauss-Krüger, 3rd meridian, Bessel ellip. -->
          <gml:pos srsDimension="3">3603845.54 5793898.77 44.8</gml:pos>
        </gml:Point>
      </referencePoint>
    </ImplicitGeometry>
  </lod2ImplicitRepresentation>
</CityFurniture>
```

The shape of the geometry of the traffic light (city furniture with class “1080” and function “1000” according to the external code lists proposed in annex A) is defined by a VRML file which is specified by a URL. This library object, which is defined in a local coordinate system, is transformed to its actual location by adding the coordinates of the reference point.

The following clip of a CityGML file provides a more complex example for an implicit geometry:

```xml
<CityFurniture>
  <class>1080</class> <!-- “traffic light”; declared in external code list (CityFurnitureClassType) in annex A -->
  <function>1000</function> <!-- “traffic”; declared in external code list (CityFurnitureFunctionType) in annex A -->
  <lod2ImplicitRepresentation>
    <ImplicitGeometry>
      <mimeType>model/vrml</mimeType>
      <transformationMatrix>
        0.866025 -0.5 0 0.7
        0.5 0.866025 0 0.8
        0 0 1 0
        0 0 0 1
      </transformationMatrix>
      <libraryObject>
        http://www.some-3d-library.com/3D/furnitures/TrafficLight434.wrl
      </libraryObject>
      <referencePoint>
        <gml:Point srsName="EPSG:31467"> <!-- Gauss-Krüger, 3rd meridian, Bessel ellip. -->
          <gml:pos srsDimension="3">3603845.54 5793898.77 44.8</gml:pos>
        </gml:Point>
      </referencePoint>
    </ImplicitGeometry>
  </lod2ImplicitRepresentation>
</CityFurniture>
```
In addition to the first example, a transformation matrix is specified. It is a homogeneous matrix, serialized in a row major fashion, i.e. the first four entries in the list denote the first row of the matrix, etc. The matrix combines a translation by the vector \((0.7, 0.8, 0)\) – the origin of the local reference system is not the center of the object – and a rotation around the z-axis by 30 degrees \((\cos(30) = 0.866025 \text{ and } \sin(30) = 0.5)\). This rotation is necessary to align the traffic light with respect to a road. The actual position of the traffic light is computed as follows:

1. each point of the VRML file (with homogeneous coordinates) is multiplied by the transformation matrix;
2. for each resulting point, the reference point \((3603845.54, 5793898.77, 44.8, 1)^T\) is added, yielding the actual geometry of the city furniture.

**External code lists**

The ImplicitGeometry model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and chapter 11.1):

- **MimeTypeType**
9 Thematic model

The thematic model of CityGML consists of the class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or be important in many different application areas in the sense of a core model. Most thematic classes are (transitively) derived from the basic classes Feature and FeatureCollection, the basic notions defined in ISO 19109 and GML3 for the representation of spatial objects and their aggregations. Features contain spatial as well as non-spatial attributes which are mapped to GML3 feature properties with corresponding data types. Geometric properties are represented as associations to the geometry classes described in chapter 8. The thematic model also comprises different types of interrelationships between Feature classes like aggregations, generalizations and associations.

The aim of the explicit modeling is to reach a high degree of semantic interoperability between different applications. By specifying the thematic concepts and their semantics along with their mapping to UML and GML3 different applications can rely on a well-defined set of Feature types, attributes, and data types with a standardized meaning or interpretation. In order to allow also for the exchange of objects and/or attributes that are not explicitly modelled in CityGML, the concepts of GenericCityObjects and GenericAttributes have been introduced (see chapter 9.10).

9.1 Top level classes

Fig. 14 presents the top level class hierarchy of the thematic model in CityGML. The base class of all thematic classes is CityObject, which provides a creation and a termination date for the management of histories of features as well as external references to the same object in other data sets (XML schema definition see chapter 10.1.2). CityObject is a subclass of the GML class Feature, thus it inherits metadata (e.g. information about the lineage, quality aspects, accuracy) and names from Feature and its super classes. A CityObject may have multiple names, which are optionally qualified by a so-called codeSpace. This enables to distinguish, for example, an official name from a popular name or names in different languages (c.f. the name property of GML objects, Cox et al. 2004).

Fig. 14: UML diagram of the top level class hierarchy of CityGML. The bracketed numbers following the attribute names denote its multiplicity: the minimal and maximal number of occurrences of the attribute per object. For example, a name is optional (0) in the class Feature or may occur multiple times (star symbol), while a CityObject has none or at most one creationDate. A ReliefFeature has exactly one occurrence of the attribute LOD.
The subclasses of CityObject comprise the different thematic fields of a city model: the terrain, the coverage by
land use objects, transportation, vegetation, water bodies and sites, in particular buildings. The class CityFur- 
niture is used to represent traffic lights, traffic signs, flower buckets, or similar objects. The class GenericCityOb-
ject allows to model features, which are not covered explicitly by the CityGML schema. In the future, sites will
be completed by further subclasses like tunnel, bridge, excavation, wall or embankment. At present, these
objects have to be represented by GenericObjects (see chapters 7.8 and 9.10).

Thematic classes 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 multiple times. The class attribute
describes the classification of the objects, e.g. road, track, railway, or square. The attribute function contains the
purpose of the object, like national highway or county road, while the attribute usage may define, if an object is
e.g. navigable or usable for pedestrians.

_CityObjectType, _CityObject

```
<xs:complexType name="_CityObjectType" abstract="true"> 
  <xs:complexContent name="_CityObject" abstract="true"> 
    <xs:restriction base="gml:AbstractFeatureType"> 
      <xs:sequence> 
        <xs:element name="creationDate" type="xs:date" minOccurs="0" /> 
        <xs:element name="terminationDate" type="xs:date" minOccurs="0" /> 
        <xs:element name="references" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded" /> 
      </xs:sequence> 
    </xs:restriction> 
  </xs:complexContent> 
</xs:complexType>
```

The generalization relation may be used to relate features, which represent the same real-world object in differ-
ent Levels-of-Detail, i.e. a feature and its generalized counterpart(s). The direction of this relation is from the
feature to the corresponding generalized feature.

Every CityObject may be assigned an arbitrary number of generic attributes. Further details on generic attributes
(including the UML diagram) are given in chapter 9.10.

CityModelType, CityModel

```
<xs:complexType name="CityModel"> 
  <xs:complexContent base="gml:AbstractFeatureCollectionType"> 
    <xs:sequence /> 
  </xs:complexContent> 
</xs:complexType>
```

_SiteType, _Site

```
<xs:complexType name="_SiteType" abstract="true"> 
  <xs:complexContent base="_CityObject" /> 
</xs:complexType>
```
_Site is intended as the abstract superclass for buildings, facilities, etc. Future extension of CityGML (e.g. bridges or tunnels) 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, generic attributes and generalization relations.

**ExternalReferenceType**

```xml
<xs:complexType name="ExternalReferenceType">
  <xs:sequence>
    <xs:element name="informationSystem" type="xs:anyURI" minOccurs="0" />
    <xs:element name="externalObject" type="ExternalObjectReferenceType" />
  </xs:sequence>
</xs:complexType>
```

An *ExternalReference* defines a hyperlink to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the British 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.

**GeneralizationRelationType**

```xml
<xs:complexType name="GeneralizationRelationType">
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_CityObject" />
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```
9.2 Digital Terrain Model (DTM)

An essential part of a city model is the terrain. In CityGML, the terrain is represented by the class ReliefFeature in LOD 0-4 (Fig. 15 depicts the UML diagram, for XML schema definition see chapter 10.3.7). A ReliefFeature 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 derivatives of CityObject, the corresponding attributes and relations are inherited. The class ReliefComponent is associated 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, TINRelief), by break lines (BreaklineRelief), or by mass points (MasspointRelief). The four types are implemented by the corresponding GML3 classes: grids by RectifiedGridCoverages, break lines by Curves, mass points by Points and TINs either by TriangulatedSurfaces or by GML3 TINs. In case of TriangulatedSurfaces, the triangles are given explicitly while in case of GML3 TINs, 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.

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. 16). 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 modeling 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 extend attributes of the corresponding terrain objects.
Accuracy and resolution of the DTM are not necessarily dependent on those of the building model. 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 (see chapter 7.4). The TIC can be used like break lines to adjust the DTM to different building models and hence to assure a consistent representation of the DTM. If necessary, a retriangulation has to be processed. A TIC can also be derived by the individual intersection of the DTM and the building model.

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 e.g. for a LOD 3 scene it might be sufficient to use a regular grid in LOD 2 with certain higher precision areas defined by ReliefComponents in LOD 3. The LOD 2 grid and the LOD 3 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 LOD 3 models which is indicated by setting its lod value to 3.

ReliefFeatureType, ReliefFeature

_ReliefComponentType, _ReliefComponent
The geometry of a TINRelief is defined by the GML geometry class gml:TriangulatedSurface. This allows to either explicitly provide a set of triangles (gml:TriangulatedSurface) or to specify 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 application of a constrained Delaunay triangulation algorithm (cf. Okabe et al. 1992).
BreaklineReliefType, BreaklineRelief

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.
9.3 Building model

The building model is the most detailed thematic concept of CityGML. It allows the representation of thematic and spatial aspects of buildings, building parts and installations in four levels of detail, LOD1 to LOD4. Fig. 17 provides examples of 3D city models for each LOD.

Fig. 17: 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. 18 and Fig. 19, for XML schema definition see chapter 9.3.1 and following. 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 specialized 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 realized. _AbstractBuilding is a subclass of the root class _CityObject. Thus, the relation to the class ExternalReference (see chapter 7.6) and the possibility to assign GenericAttributes is inherited (see chapter 7.8).

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 (see chapter 7.7). The main building of the complex can be denoted by providing “main building” as the role name of the corresponding object.

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 roof type, the measured height, and the number and individual heights of the stories 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, Addresses can be assigned to Buildings or BuildingParts.
The geometric representation and semantic structure of an AbstractBuilding is shown in Fig. 18. The model is successively refined from LOD1 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 minimum acquisition criteria for each LOD (cf. chapter 7.1). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

Fig. 18: UML diagram of CityGML’s building model, part 1: pivotal class AbstractBuilding, Room, and thematic surfaces.

Fig. 19: UML diagram of CityGML’s building model, part 2: Building, BuildingPart, BuildingInstallations, BuildingFurniture, and Address.
In LOD1, a building model consists of a geometric representation of the building volume. Optionally, a MultiCurve representing the TerrainIntersectionCurve (see chapter 7.4) can be specified. This geometric representation is refined in LOD2 by additional MultiSurface and MultiCurve geometries, used for modeling architectural details like a roof overhang, columns, or antennas. In LOD2 and higher LODs, the outer façade of a building also can be differentiated semantically by the classes _BoundarySurface and BuildingInstallation. A _BoundarySurface is a part of the building’s exterior shell with a special function like wall (WallSurface), roof (RoofSurface), ground plate (GroundSurface) 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 (c.f. Fig. 19).

In LOD3, the openings in _BoundarySurface objects (doors and windows) can be represented as thematic objects. 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 (see chapter 7.7). A Room may have the attributes class, function and usage, referencing to external code lists (chapter 9.3.7 and 11.1). 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 specialized _BoundarySurfaces, representing floor (FloorSurface), ceiling (CeilingSurface), and interior walls (InteriorWallSurface). Installations like lamps or radiators, as well as room furniture like tables and chairs, can also be represented in the CityGML building model. For the immovable equipment of a room like pillars or stairs, the class BuildingInstallations has to be used, while movable furniture can be specified with the class BuildingFurniture. A BuildingFurniture may have the attributes class, function and usage.

All building models from LOD 1 to 4 can have individual or generic TexturedSurfaces, which provide materials (e.g. colors) and textures for visualization, which vary in resolution depending on the LOD (see chapter 8.2). An example of a CityGML dataset for SimpleBuildings can be found in chapter 11.4.1.

9.3.1 Building and building part

BuildingType, Building

```
<xs:complexType name="BuildingType">
  <xs:complexContent>
    <xs:extension base="_AbstractBuildingType">
      <xs:sequence/>
      <xs:extension base="_AbstractBuilding"/>
      <xs:attribute name="Building" type="BuildingType" substitutionGroup="_AbstractBuilding"/>
    </xs:complexType>
  </xs:complexContent>
</xs:complexType>
```

The Building class is one of the two subclasses of AbstractBuilding. If a building only consists of one (homogeneous) part, this class shall be used. A building composed of structural segments differing in e.g. the number of storeys or the roof type has to be separated into one Building having one or more additional BuildingParts (see Fig. 20). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating Building feature.

BuildingPartType, BuildingPart

```
<xs:complexType name="BuildingPartType">
  <xs:complexContent>
    <xs:extension base="_AbstractBuildingType"/>
  </xs:complexContent>
</xs:complexType>
```

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved. 33
The class BuildingPart is derived from AbstractBuilding. It is used to model a structural part of a building (see Fig. 20).

Fig. 20: Examples of buildings consisting of one and two building parts (source: City of Coburg).
The abstract class _AbstractBuilding contains properties for building attributes, purely geometric representations, and geometric/semantic representations of the building or building part on different levels of detail. The attributes describe:

- The different functions of the building or building part (function). The allowed values for the BuildingFunctionType are specified in a separate XML file, using the dictionary concept of GML3.
- The year of construction of the building or building part (yearOfConstruction).
- The roof type of the building or building part (roofType). The allowed values for the RoofTypeType are specified in a separate XML-File, using the dictionary concept of GML.
- The measured relative height (measuredHeight) of the building or building part ridge line (highest point).
- The number of storeys above (storeyAboveGround) and below (storeyBelowGround) ground level.
- 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 geometric representation and the thematic structuring of the model into components with a special semantic meaning. This is illustrated in Fig. 21, showing the same building in four different LODs. The class _AbstractBuilding has a number of properties which are associated with certain LODs.

![Fig. 21: Building model in LOD1 – LOD4 (source: Research Center Karlsruhe).](image)

Tab. 2 (see next page) shows the correspondence of the different geometric and semantic themes of the building model to LODs. In each LOD, the volume of a building can be expressed by a SolidGeometry and/or a MultiSurfaceGeometry. 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 all four LODs. The TIC can – but does not have to – build closed rings around the building or building parts.

In LOD1 (see Fig. 21 a), the different structural entities of a building are aggregated to simple blocks 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 SolidGeometry and/or MultiSurfaceGeometry, it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is _BoundarySurface (see chapter 9.3.2), which is associated with a MultiSurfaceGeometry. If in a building model both a geometric representation of the exterior shell as
volume or surface model and a semantic representation by means of thematic BoundarySurfaces exist, the geometric representation must not explicitly define the geometry, but has to reference the MultiSurfaceGeometry of the BoundarySurfaces.

<table>
<thead>
<tr>
<th>Geometric / semantic theme</th>
<th>Property type</th>
<th>LOD 1</th>
<th>LOD 2</th>
<th>LOD 3</th>
<th>LOD 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume part of the building shell</td>
<td>gml:SolidType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Surface part of the building shell</td>
<td>gml:MultiSurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Terrain Intersection Curve</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Curve part of the building shell</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>BoundarySurfaces (chapter 9.3.2)</td>
<td>BoundarySurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Building installations (chap. 9.3.3)</td>
<td>BuildingInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Openings (chapter 9.3.4)</td>
<td>OpeningType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Rooms (chapter 9.3.5)</td>
<td>InteriorRoomType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

Tab. 2: Semantic themes of the class AbstractBuilding.

Besides BuildingParts, also smaller features of the building can strongly affect the building characteristic. These features are modelled by the class BuildingInstallation (see chapter 9.3.3). Typical candidates for this class are chimneys (see Fig. 21 c), dormers (see Fig. 20), balconies, outer stairs, or antennas. BuildingInstallations may only be included in LOD2 models, if their extents exceed the minimum dimensions as specified in chapter 7.1. For the geometrical representation of a BuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used.

The class AbstractBuilding has no additional properties for LOD3. Besides the higher requirements on geometric precision and smaller minimum dimensions, the main difference of LOD2 and LOD3 buildings concerns the class BoundarySurface (see chapter 9.3.2). In LOD3, openings in a building corresponding with windows or doors (see Fig. 21 c) are modeled by the (abstract) class Opening and the derived classes Window and Door (see chapter 9.3.4).

With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. But LOD4 provides for a possibility to describe the interior structure of a building with the class Room (see chapter 9.3.5).

AddressType, Address

Each Building or BuildingPart feature may be assigned zero or more addresses. Addresses are modeled as GML features having one address property and an optional multiPoint property, which allows specification of the exact positions of the building entrances that are associated with the corresponding address. The point coordinates can be 2D or 3D. Modeling addresses as features has the advantage that GML3’s method of representing features by reference (using XLinks) can be applied. This means, that addresses might be bundled as an address FeatureCol-
lection 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, the structure of the child element of the xALAddress property has to follow the OASIS xAL schema. Listing 1 and Listing 2 give examples for the representation of German and British addresses in xAL. 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:xsd:schema:xAL:2.0”. An example showing a complete CityGML building including an address element is provided in chapter 11.4.1.

Listing 1: Example for a German address in xAL format.

```xml
<!-- Bussardweg 7, 76356 Weingarten, Germany -->
<xal:AddressDetails>
  <xal:Country>
    <xal:CountryName>Germany</xal:CountryName>
  </xal:Country>

  <xal:Locality Type="City">
    <xal:LocalityName>Weingarten</xal:LocalityName>
    <xal:Thoroughfare Type="Street">
      <xal:ThoroughfareName>Bussardweg</xal:ThoroughfareName>
      <xal:ThoroughfareNumber>7</xal:ThoroughfareNumber>
    </xal:Thoroughfare>
    <xal:PostalCode>
      <xal:PostalCodeNumber>76356</xal:PostalCodeNumber>
    </xal:PostalCode>
  </xal:Locality>
</xal:AddressDetails>
```

Listing 2: Example for a british address in xAL format (source: http://xml.coverpages.org/xnal.html).

```xml
<!-- 46 Brynmaer Road Battersea LONDON, SW11 4EW United Kingdom -->
<xal:AddressDetails>
  <xal:Country>
    <xal:CountryName>United Kingdom</xal:CountryName>
  </xal:Country>

  <xal:Locality Type="City">
    <xal:LocalityName>LONDON</xal:LocalityName>
    <xal:DependentLocality Type="District">
      <xal:DependentLocalityName>BATTERSEA</xal:DependentLocalityName>
    </xal:DependentLocality>
    <xal:Thoroughfare>
      <xal:ThoroughfareName>BRYNMAER ROAD</xal:ThoroughfareName>
      <xal:ThoroughfareNumber>46</xal:ThoroughfareNumber>
    </xal:Thoroughfare>
    <xal:PostalCode>
      <xal:PostalCodeNumber>SW11 4EW</xal:PostalCodeNumber>
    </xal:PostalCode>
  </xal:Locality>
</xal:AddressDetails>
```

9.3.2 Boundary surfaces

_BoundarySurfaceType, _BoundarySurface

_BoundarySurfaceType is the (abstract) base class for several semantic classes, structuring the exterior shell of a building and the visible surface of a room. 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 GenericAttributes and ExternalReferences. From _BoundarySurface, the thematic classes RoofSurface, WallSurface, GroundSurface, ClosureSurface, FloorSurface, InteriorWallSurface, and CeilingSurface are derived. The thematic classification of building surfaces is illustrated in Fig. 22 and specified subsequently.
For each LOD between 2 and 4, the geometry of a _BoundarySurface may be defined by a different MultiSurfaceGeometry. In LOD2, this surface geometry must be simply connected, which means that the components of the MultiSurface (e.g. gml:Polygon) must not have inner holes (gml:interior).

In LOD3 and LOD4, a _BoundarySurface may reference Openings (see 9.3.4) like doors and windows. If the geometric location of Openings lies topologically within a surface component (e.g. gml:Polygon) of the MultiSurfaceGeometry, these Openings must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. 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. However, the points have to be specified in reverse order (exterior boundaries clockwise and interior boundaries clockwise when looking in opposite direction of the surface’s normal vector). The embrasure surfaces of an Opening belong to the according adjacent _BoundarySurface. If e.g. 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. 22 on the right).

Fig. 22: Classification of BoundarySurfaces (left), in particular for Openings (right) (graphic: IKG Uni Bonn).

**RoofSurfaceType, RoofSurface**

```
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="_BoundarySurfaceType" />
  </xs:complexContent>
</xs:complexType>
```

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 BuildingInstallations.

**WallSurfaceType, WallSurface**

```
<xs:complexType name="WallSurfaceType">
  <xs:complexContent>
    <xs:extension base="_BoundarySurfaceType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

All parts of the building facade visible from the outside are modelled by the class WallSurface.

**GroundSurfaceType, GroundSurface**

```
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="_BoundarySurfaceType" />
  </xs:complexContent>
</xs:complexType>
```
The ground plate of a building or building part is modelled by the class \textit{GroundSurface}.

\textbf{ClosureSurfaceType, ClosureSurface}

An opening in a building not filled by a door or window can be sealed by a virtual surface called \textit{ClosureSurface} (see chapter 7.3). Hence, buildings with open sides like a barn or a hangar, can be virtually closed in order to be able to compute their volume. \textit{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 \textit{ClosureSurface} should be used to separate or connect the volumes of both rooms.

\textbf{FloorSurfaceType, FloorSurface}

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

\textbf{InteriorWallSurfaceType, InteriorWallSurface}

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

\textbf{CeilingSurfaceType, CeilingSurface}

The class \textit{CeilingSurface} must only be used in the LOD4 interior building model for modelling the ceiling of a room.
9.3.3 BuildingInstallations

BuildingInstallationType, BuildingInstallation

```xml
<xs:complexType name="BuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="BuildingInstallationClassType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="function" type="BuildingInstallationFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="BuildingInstallationUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A BuildingInstallation is a part 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 has an (optional) function describing the semantic meaning of the installation. The list of feasible functions for BuildingInstallations is specified in a GML dictionary. For the geometrical representation of a BuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 8 can be used.

9.3.4 Openings

_OpeningType, _Opening

```xml
<xs:complexType name="_OpeningType" abstract="true">
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The class _Opening is the (abstract) base class for semantically describing openings like doors or windows in outer or inner walls. Openings only exist in models of LOD3 or LOD4. Each _Opening is associated with a MultiSurfaceGeometry.

WindowType, Window

```xml
<xs:complexType name="WindowType">
  <xs:complexContent>
    <xs:extension base="_OpeningType">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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.
DoorType, Door

```xml
<xs:complexType name="DoorType">
  <xs:complexContent>
    <xs:extension base="_OpeningType">
      <xs:sequence>
        <xs:element name="address" type="AddressPropertyType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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 a 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.

9.3.5 Building Interior

RoomType, Room

```xml
<xs:complexType name="RoomType">
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="RoomClassType" minOccurs="0" />
        <xs:element name="function" type="RoomFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="RoomUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0" />
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="interiorBuildingInstallation" type="BuildingInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

A Room is a semantic object for modelling the free space inside a building. 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 (use the orientation attribute of the appearance property element).

In addition to the geometrical representation, different parts of the visible surface of a room can be modelled by specialized BoundarySurfaces (FloorSurface, CeilingSurface, InteriorWallSurface, see chapter 9.3.2).

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 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. 23).
Fig. 23: Accessibility graph derived from topological adjacencies of room surfaces (graphic: IKG Uni Bonn).

**BuildingFurnitureType, BuildingFurniture**

```xml
<xs:complexType name="BuildingFurnitureType">
    <xs:complexContent>
        <xs:extension base="_CityObjectType">
            <xs:sequence>
                <xs:element name="class" type="BuildingFurnitureClassType" minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="function" type="BuildingFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="usage" type="BuildingFurnitureUsageType" minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="lod4ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Rooms may have BuildingFurnitures and interior BuildingInstallations. A BuildingFurniture is a movable part of a room, such as a chair or furniture while a BuildingInstallation (see 9.3.3) is permanently connected to the room like stairs or pillars. BuildingFurniture is modeled in the same way as CityFurniture (cf. chapter 9.7).

### 9.3.6 Modelling further building features using CityObjectGroups

CityGML does not provide a specific concept for the representation of storeys as it is available in the AEC/FM standard IFC (IAI 2005). However, a storey can be represented as an explicit aggregation of all building features on a certain height level using CityGML’s notion of CityObjectGroups (cf. chapter 9.9). This would include Rooms, Doors, Windows, BuildingInstallations and BuildingFurniture. If thematic surfaces like walls and interior walls should also be associated to a specific storey, this might require the vertical fragmentation of these surfaces (one per storey), as in virtual 3D city models they typically span the whole façade.

By proceeding this way, the storey-wise grouping of entities can be preserved, if IFC building models should be converted or mapped to CityGML. In order to express that this group belongs to the building theme, the class property of the corresponding CityObjectGroup shall be assigned the value “building separation”. The function property shall be assigned the value “lodXStorey” with x between 1 and 4 in order to denote that this group represents a storey wrt. a specific LOD. The storey name or number can be stored in the gml:name property of the CityObjectGroup feature.

### 9.3.7 External code lists

The building model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- BuildingClassType
- BuildingFunctionType
• BuildingUsageType
• RoofTypeType
• BuildingInstallationClassType
• BuildingInstallationFunctionType
• BuildingInstallationUsageType
• BuildingFurnitureClassType
• BuildingFurnitureFunctionType
• BuildingFurnitureUsageType
• RoomClassType
• RoomFunctionType
• RoomUsageType
9.4 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 sanitation. 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 yield in high-tech bridges of today. The skylines of many cities are dominated by impression of water, which directly relates to 3D city models. Further on, 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 steady buildings, roadways, and terrain is considered in this thematic model. Water bodies are dynamic surfaces. Tides occur regularly, but non-periodic changes predominate with respect to elemental forces. 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. But the semantic and geometric description given here allows further enhancements of dynamics and conceptually different descriptions.

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. 24). A dynamic element may be the WaterSurface to represent temporarily changing situations of tidal flats.

![Fig. 24: Illustration of a water body defined in CityGML (graphic: IKG Uni Bonn).](image)

The UML diagram of the water body model is depicted in Fig. 25, for XML schema definition see below and chapter 10.3.5.

Every WaterBody object may have the attributes class, function and usage, referencing to external code lists (c.f. chapter 9.4.3 and 11.1). 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 e.g. 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 thus of the root class _CityObject. The class _WaterObject can 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 External-References and GenericAttributes (c.f. chapter 7.6, 7.8).

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 are defined in the respective external code list.

Fig. 25: UML diagram of the water body model in CityGML.

LOD0 and LOD1 represent a low level of illustration and high grade of generalization. Here the rivers are modelled as MultiCurve geometry and creeks are omitted. Seas, oceans and lakes with significant extent are represented as a MultiSurface (Fig. 25). 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 geometries 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 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 Solid. This can be ensured, if for one LOD X the respective lodXSurface elements (where X is between 2 and 4) of WaterClosureSurface, WaterGroundSurface, and WaterSurface do not redundantly define gml:Polygons, but instead reference the corresponding polygons (using XLink) within the CompositeSurface that defines the exterior shell of the Solid.

LOD2 to LOD4 demand a higher grade of detail and therefore any WaterBody will be outlined by thematic surfaces or a solid composed of the surrounding thematic surfaces.

The dashed lines mean that there exists one association for each of the listed LOD. Every object of the class WaterSurface, WaterClosureSurface, and WaterGroundSurface must have at least one associated surface geometry. This means, that every WaterSurface, WaterClosureSurface, and WaterGroundSurface 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 7.4). The rings defining the WaterSurface polygons implicitly delineate the intersection of the water body with the terrain or basin.

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
9.4.1 Water body

_WaterObjectType, _WaterObject

<x:complexType name="_WaterObjectType" abstract="true">
  <x:complexContent>
    <x:extension base="_CityObjectType" />
  </x:complexContent>
</x:complexType>

WaterBodyType, WaterBody

<x:complexType name="WaterBodyType">
  <x:complexContent>
    <x:extension base="_WaterObjectType">
      <x:sequence>
        <x:element name="class" type="WaterBodyClassType" minOccurs="0" />
        <x:element name="function" type="WaterBodyFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <x:element name="usage" type="WaterBodyUsageType" minOccurs="0" maxOccurs="unbounded" />
        <x:element name="lod0MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0" />
        <x:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <x:element name="lod1MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0" />
        <x:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <x:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0" />
        <x:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0" />
        <x:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0" />
        <x:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0" />
        <x:element name="boundedBy" type="BoundedByWaterSurfacePropertyType" minOccurs="0" maxOccurs="unbounded" />
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x:complexType>

9.4.2 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. 24).

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 changing water levels, which can be reflected by including different static water surfaces having different water-Levels (WaterLevelType), as e.g. highest flooding event, mean sea level, or minimum water level, given in an external code list. 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 Building”. The ground surface might be identical to the underwater terrain model, but also describes the contour to other underwater objects. The usefulness of this concept arises from the existence of water defence buildings like sluices, sills, flood barrage or tidal power stations. The use of WaterGroundSurface as boundary layer to buildings is relevant in urban situations, where buildings enclose the defined water completely such as fountains and swimming pools. Together, the WaterSurface and WaterGroundSurface enclose to 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 between water volumes from such, 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 shall only be included as parts of corresponding WaterBody objects and may not be used as stand-alone objects within a CityGML model.

_WaterBoundarySurfaceType, _WaterBoundarySurface

WaterSurfaceType, WaterSurface

WaterGroundSurfaceType, WaterGroundSurface

WaterClosureSurfaceType, WaterClosureSurface

9.4.3 External code lists

The water bodies model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- WaterLevelType
- WaterBodyClassType
- WaterBodyFunctionType
- WaterBodyUsageType
9.5 Transportation objects

The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and functional 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 to apply geometric route planning algorithms. This can be useful to determine constrictions and needed manoeuvres 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 visualizations and training simulators.

The main class is **TransportationComplex**, which represents, for example, a road, a track, a railway, or a square. Fig. 26 illustrates the four different thematic classes.

A **TransportationComplex** is composed of the parts **TrafficArea** and **AuxiliaryTrafficArea**. Fig. 27 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 flower buckets.

Fig. 28 depicts the UML diagram of the transportation model, for XML schema definition see chapter 10.3.3.
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 the car driving lanes, the pedestrian zones and cyclists zones. The AuxiliaryTrafficAreas are describing further elements of the road, like kerbstones, road markings and grass stripes.

TransportationComplex objects can be thematically differentiated using the subclasses Track, Road, Railway, and Square. Every TransportationComplex has the attributes function and usage, referencing to external code lists (chapter 9.5.3 and 11.1). The attribute function describes the purpose of the object like e.g. national motorway, country road, or airport, while the attribute usage can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

In addition every TrafficArea may have the attributes function, usage, and surfaceMaterial. The function describes, if the object may be a car driving lane, a pedestrian zones, or a cyclists zone, while the usage attribute indicates which modes of transportation it is used (e.g. pedestrian, car, tram, roller skates). The attribute surfaceMaterial specifies the type of pavement and may also be used for AuxiliaryTrafficAreas (e.g. asphalt, concrete, gravel, soil, rail, grass etc.). The function attribute of the AuxiliaryTrafficArea defines e.g. kerbstones, road markings, or grass stripes. The possible values are also specified in external code lists.

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

<table>
<thead>
<tr>
<th>Example:</th>
<th>Country road</th>
<th>Motorway entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransportationComplex</td>
<td>road</td>
<td>road</td>
</tr>
<tr>
<td>(Aux-) TrafficArea - Usage</td>
<td>car, truck, bus, taxi, motorcyclist</td>
<td>car, truck, bus, taxi, motorcyclist</td>
</tr>
<tr>
<td>(Aux-) TrafficArea - Function</td>
<td>driving lane</td>
<td>motorway entry</td>
</tr>
<tr>
<td>(Aux-) TrafficArea - SurfaceMaterial</td>
<td>asphalt</td>
<td>concrete</td>
</tr>
</tbody>
</table>
TransportationComplex is a subclass of TransportationObject and of the root class CityObject. The geometrical representation of the TransportationComplex varies through the different levels of detail. Since TransportationComplex is a subclass of CityObject and hence a feature, it inherits the attribute gml:name. As well the street name is stored within the gml:name property of the Road feature.

In the coarsest LOD0 the transportation complexes are modeled 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 visualizations of the transport network. Since this abstract definition of transportation network does not contain explicit description of the transportation objects, it may be task of the viewer application to generate the graphical visualization, 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 means like 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 illustrated in Fig. 29.

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, GenericAttributes can be used or further objects of interest can be added from other information systems by the use of ExternalReferences (see chapter 7.8 and 7.6). For example, GDF data-sets, which provide additional information for car navigation, can be used for simulation and visualization of traffic flows. The values of the object attributes can be augmented using the concept of dictionaries (see chapter 7.5). These directories may be country- or user-specific (especially for country-specific road signs and signals).
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.

9.5.1 Transportation complex

_TransportationObjectType, _TransportationObject

```
<xs:complexType name="_TransportationObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="_CityObjectType" />
  </xs:complexContent>
</xs:complexType>
```

```
<xs:element name="_TransportationObject" type="_CityObjectType" substitutionGroup="_CityObject" />
```
OGC 06-057

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

**TransportationComplexType, TransportationComplex**

This type and element describes transportation complexes like roads or railways which may be aggregated of 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, generic attributes and generalization relations. Furthermore, it represents the superclass for thematically distinct types of transportation complexes.

**TrackType, Track**

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**

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

```
<x:s:complexType name="RailwayType">
  <x:s:complexContent>
    <x:s:extension base="TransportationComplexType">
      <x:s:sequence />
    </x:s:extension>
  </x:s:complexContent>
</x:s:complexType>
```

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

SquareType, Square

```
<x:s:complexType name="SquareType">
  <x:s:complexContent>
    <x:s:extension base="TransportationComplexType">
      <x:s:sequence />
    </x:s:extension>
  </x:s:complexContent>
</x:s:complexType>
```

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

9.5.2 Subclasses of transportation complexes

TrafficAreaType, TrafficArea

```
<x:s:complexType name="TrafficAreaType">
  <x:s:complexContent>
    <x:s:extension base="_TransportationObjectType">
      <x:s:sequence>
        <x:s:element name="usage" type="TrafficAreaUsageType" minOccurs="0" maxOccurs="unbounded"/>
        <x:s:element name="function" type="TrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <x:s:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" minOccurs="0" maxOccurs="unbounded"/>
        <x:s:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
        <x:s:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
        <x:s:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
      </x:s:sequence>
    </x:s:extension>
  </x:s:complexContent>
</x:s:complexType>
```

AuxiliaryTrafficAreaType, AuxiliaryTrafficArea

```
<x:s:complexType name="AuxiliaryTrafficAreaType">
  <x:s:complexContent>
    <x:s:extension base="_TransportationObjectType">
      <x:s:sequence>
        <x:s:element name="function" type="AuxiliaryTrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <x:s:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" minOccurs="0" maxOccurs="unbounded"/>
        <x:s:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
        <x:s:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
        <x:s:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="0"/>
      </x:s:sequence>
    </x:s:extension>
  </x:s:complexContent>
</x:s:complexType>
```
9.5.3 External code lists

The transportation model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- TransportationComplexFunctionType
- TransportationComplexUsageType
- TrafficAreaFunctionType
- TrafficAreaUsageType
- AuxiliaryTrafficAreaFunctionType
- SurfaceMaterialType
9.6 Vegetation objects

Vegetation features are important components of a 3D city model, since they support the recognition of the surrounding environment. By the analysis and visualisation of vegetation objects, statements on their distribution, structure and diversification can be made. Habitats can be analysed and impacts on the fauna can be derived. The vegetation model may be used as a basis for simulations of e.g. forest fire, urban aeration or microclimate. The model could be used e.g. to examine forest damage, to detect obstacles (e.g. concerning air traffic) or to perform analysis tasks in the field of environmental protection.

The vegetation model of CityGML distinguishes between solitary vegetation objects like trees and vegetation areas, which represent biotopes like forests or other plant communities (Fig. 32). Single vegetation objects are modelled by the class SolitaryVegetationObject, while for areas filled with a specific vegetation the class PlantCover is used. The geometry representation of a PlantCover feature may be a MultiSurface or a MultiSolid, depending on the vertical extend of the vegetation. For e.g. forests, a MultiSolid representation might be more appropriate. The UML diagram of the vegetation model is depicted in Fig. 33, for XML schema definition see below or chapter 10.3.4.

![Image of vegetation objects](graphic: District of Recklinghausen)

A SolitaryVegetationObject may have the attributes class, species, function, height, trunkDiameter and crownDiameter. The attribute class contains the coarse classification of the object or plant habit, e.g. tree, bush, grass, and can occur only once (see external code list in chapter 9.6.4 and 11.1). The attribute species defines the species’ name like e.g. “Abies alba”, and can occur at most once (see external code list in chapter 9.6.4 and 11.1). The hierarchy between class and species is not reflected in the external code lists, thus inconsistencies have to be checked by application tools. The optional attribute function denotes the purpose of the object like e.g. botanical monument, and can occur multiple times. The attribute height contains the relative height of the object. The attributes crownDiameter and trunkDiameter represent the plant crown resp. trunk diameter. The trunk diameter is often used in regulations of municipal cadastre (e.g. tree management rules).

A PlantCover feature may have the attributes class, function and averageHeight. The plant community of a PlantCover is represented by the attribute class. The values of this attribute are enumerated in an external code list (chapter 9.6.4 and 11.1), where each value describes not only one plant type or species, but denotes a typical mixture of plant types in a plant community. This information can be used in particular to generate realistic 3D visualisations, where the PlantCover region is automatically, perhaps randomly, filled with a corresponding mixture of 3D plant objects. The attribute function indicates the purpose of the object like e.g. national forest, and can occur multiple times. The attribute averageHeight denotes the average relative vegetation height.

Since a SolitaryVegetationObject and a PlantCover is a CityObject, it inherits all attributes of a city object, in particular a name (gml:name) and an ExternalReference to a corresponding object in an external information system, which may contain botanical information from public environmental agencies (see chapter 7.6).
The geometry of a SolitaryVegetationObject may be defined in LOD 1-4 explicitly by a GML geometry having absolute coordinates, or prototypically by an ImplicitGeometry (see chapter 8.3). Solitary vegetation objects probably are one of the most important features where implicit geometries are appropriate, since the shape of the most types of vegetation objects, e.g. trees of the same species, can be treated as identical in most cases. Furthermore, season dependent appearances may be mapped using ImplicitGeometries. For visualization purposes, only the content of the library object defining the object’s shape and appearance has to be swapped (cf. Fig. 34).

Fig. 33: UML diagram of vegetation objects in CityGML.

Fig. 34: Visualisation of a vegetation object in different seasons (source: District of Recklinghausen).
A SolitaryVegetationObject or a PlantCover may have a different geometry in each LOD, as indicated by the dashed lines in Fig. 33. Whereas a SolitaryVegetationObject is associated with the _Geometry class representing an arbitrary GLM geometry (by the relation lodXGeometry), a PlantCover is restricted to be either a MultiSolid or a MultiSurface. An example of a PlantCover modeled as MultiSolid is a ‘solid forest model’, see Fig. 35.

![Solid forest model](image)

**Fig. 35: Example for the visualisation/modelling of a solid forest (source: District of Recklinghausen).**

### 9.6.1 Vegetation object

_VegetationObjectType, _VegetationObject

```xml
<xs:complexType name="_VegetationObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="_CityObjectType" />
  </xs:complexContent>
</xs:complexType>
```

### 9.6.2 Solitary vegetation objects

SolitaryVegetationObjectType, SolitaryVegetationObject

```xml
<xs:complexType name="SolitaryVegetationObjectType">
  <xs:complexContent>
    <xs:extension base="_VegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="PlantClassType" minOccurs="0"/>
        <xs:element name="function" type="PlantFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="species" type="SpeciesType" minOccurs="0"/>
        <xs:element name="height" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="trunkDiameter" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="crownDiameter" type="gml:LengthType" minOccurs="0"/>
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
9.6.3 Plant cover objects

PlantCoverType, PlantCover

```
<xs:complexType name="PlantCoverType">
  <xs:complexContent>
    <xs:extension base="_VegetationObjectType">
      <xs:sequence>
        <xs:element name="class" type="PlantCoverClassType" minOccurs="0" />
        <xs:element name="function" type="PlantCoverFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="averageHeight" type="gml:LengthType" minOccurs="0" />  
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod1MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
        <xs:element name="lod2MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
        <xs:element name="lod3MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

9.6.4 External code lists

The vegetation model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- PlantClassType
- PlantFunctionType
- SpeciesType
- PlantCoverClassType
- PlantCoverFunctionType

9.6.5 Example of a CityGML dataset

The following two excerpts of a CityGML dataset contain a solitary tree (SolitaryVegetationObject) and a plant community (PlantCover). The solitary tree has the attributes: class = 1070 (deciduous tree), species = 1040 (Fagus/beech), height = 8 m, trunkDiameter = 0.7 m, crownDiameter = 8.0 m. The plant community has the attributes: class = 1180 (isoeto-nanojuncetea), averageHeight = 0.5 m.

```
<SolitaryVegetationObject>
  <class>1070</class>
  <species>1040</species>
  <height uom="#m">8</height>
  <trunkDiameter uom="#m">0.7</trunkDiameter>
  <crownDiameter uom="#m">8</crownDiameter>
  <lod1ImplicitRepresentation>
    <ImplicitGeometry>
      <mimeType>1010</mimeType>
      <LibraryObject>urn:sig3d:tree.wrl</LibraryObject>
      <referencePoint>
        <gml:Point srsName="…a 3D CRS…”>
          <gml:pos srsDimension="3">2571129.123 5733690.578 60.0</gml:pos>
        </gml:Point>
      </referencePoint>
    </ImplicitGeometry>
  </lod1ImplicitRepresentation>
</SolitaryVegetationObject>
```
<PlantCover>
  <class>1040</class>
  <averageHeight uom="m">0.5</averageHeight>
  <lod1Geometry>
    <gml:Polygon srsName="...a 3D CRS...">
      <gml:exterior>
        <gml:LinearRing>
          <gml:pos srsDimension="3">2571329.227 5733806.146 60.0</gml:pos>
          <gml:pos srsDimension="3">2571387.011 5733754.782 60.0</gml:pos>
          <gml:pos srsDimension="3">2571374.170 5733674.527 60.0</gml:pos>
          <gml:pos srsDimension="3">2571274.653 5733670.246 60.0</gml:pos>
          <gml:pos srsDimension="3">2571243.621 5733706.413 60.0</gml:pos>
          <gml:pos srsDimension="3">2571329.227 5733806.146 60.0</gml:pos>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </lod1Geometry>
</PlantCover>
9.7 City furniture

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, advertising columns, benches, delimitation stakes, or bus stops (Fig. 36, Fig. 37). City furniture objects can be found in traffic areas, residential areas, on squares or in built-up areas. The modelling of city furniture objects is used for visualisation e.g. of city traffic, but also for analysing local structural conditions. The recognition of special locations in a city model is improved by the use of these detailed city furniture objects, and the city model itself becomes more alive and animated.

City furniture objects can have an important influence on simulations of e.g. city traffic situations. Navigation systems can be realised e.g. for visually handicapped people using a traffic light as routing target. Or city furniture objects are important to plan a heavy vehicle transportation, where the exact position and further conditions of obstacles must be known.

![Image](image_url)

**Fig. 36**: Real situation showing a bus stop (left). The advertising billboard and the refuge are modelled as CityFurniture objects in the right image (source: 3D city model of Barkenberg).

![Image](image_url)

**Fig. 37**: Real situation showing lanterns and delimitation stakes (left). In the right image they are modelled as CityFurniture objects with ImplicitGeometries (source: 3D city model of Barkenberg).

The UML diagram of the city furniture model is depicted in Fig. 38, for XML schema definition see below and chapter 10.3.6.

The class CityFurniture may have the attributes class and function. Their possible values are specified in the respective external code lists (chapter 9.7.2 and 11.1). The class attribute allows an object classification like traffic light, traffic sign, delimitation stake, or garbage can, and can occur only once. The function attribute describes, to which thematic area the city furniture object belongs (e.g. transportation, traffic regulation, architecture etc.), and can occur multiple times. The hierarchy between class and function is not reflected in the external code lists. Inconsistencies have to be checked by the application tools.
Since CityFurniture is a subclass of _CityObject and hence is a feature, it inherits the attribute gml:name. As with any _CityObject, CityFurniture objects may be assigned ExternalReferences and GenericAttributes (7.6, 7.8). For ExternalReferences city furniture objects can have links to external thematic databases. Therefore, semantical information of the objects, which can not be modelled in CityGML, can be transmitted and used in the 3D city model for further processing, e.g. information from systems of powerlines or pipelines, traffic sign cadaster, or water resources for disaster management.

City furniture objects can be represented in city models with its specific geometry, but in most cases the same kind of object has an identical geometry. The geometry of a CityFurniture objects in LOD 1-4 may be represented by an explicit geometry (lodXGeometry where X is between 1 and 4) or an ImplicitGeometry object (lodXImplicitRepresentation with X between 1 and 4). In the concept of ImplicitGeometry the geometry of a prototype city furniture object is stored only once in a local coordinate system and referenced by a number of features (see chapter 8.3). Spatial information of city furniture objects can be taken from city maps (called “Stadtgrundkarte” in Germany) or from public and private external information systems.

9.7.1 City furniture object

_CityFurnitureType, _CityFurniture

```xml
<xs:complexType name="CityFurnitureType">
  <xs:annotation>
    <xs:documentation>
      Type describing city furnitures, like traffic lights, benches, ... As subclass of _CityObject, a CityFurniture inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. 
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="CityFurnitureClassType" minOccurs="0" />
        <xs:element name="function" type="CityFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="lod1ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" />
        <xs:element name="lod2ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" />
        <xs:element name="lod3ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" />
        <xs:element name="lod4ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
9.7.2 External code lists

The city furniture model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- CityFurnitureFunctionType
- CityFurnitureClassType

9.7.3 Example of CityGML dataset

The following example of a CityGML dataset is an extract of the modelling of a delimitation stake in LOD3 and contains the attributes: class = 1000, function = 1520 (delimitation stake). The delimitation stake with the object ID stake0815 is referencing by urn:adv:oid:DEHE123400007001 to an cadastre object in the German ALKIS database (www.adv-online.de).

This example shows the geometry of Cover Surface (on the top of the stake) and of the left Surface left (Fig. 39). The Cover Surface has the material (colour) white and the Surface left has the texture stake.gif with according texture coordinates.

![Fig. 39: Example of a simple city furniture object (source: District of Recklinghausen).](image)
9.8 Land use

LandUse objects describe areas of the earth’s surface dedicated to a specific land use. They can be employed to represent parcels in 3D. Fig. 40 shows the UML diagram of land use objects, for the XML schema definition see chapters 9.8.1 and 10.3.7.

![UML diagram of land use objects in CityGML](image)

Every LandUse object may have the attributes class, function, and usage. The class attribute is used to represent the classification of land use objects, like settlement area, industrial area, farmland etc., and can occur only once. The possible values are specified in an external code list (see chapter 11.1). The attribute function defines the purpose of the object, like e.g. cornfield, while 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 LandUse object is defined for all LOD 0-4 and may have different geometries in any LOD. The surface geometry of a LandUse object is required to have 3D coordinate values. It must be a GML3 MultiSurface, which might be assigned material properties like textures or colours (using CityGML’s TexturedSurface).

LandUse objects can be employed to establish a coherent geometric/semantical tesselation of the earth’s surface. In this case topological relations between neighbouring LandUse objects should be made explicit by defining the boundary LineStrings only once and by referencing them in the corresponding Polygons using XLinks (cf. chapter 8.1). Fig. 41 shows a land use tesselation, where the geometries of the land use objects are represented as triangulated surfaces. In fact, they are the result of a constrained triangulation of a DTM with consideration of breaklines defined by a 2D vector map of land use classifications.

![LOD0 regional model consisting of land use objects in CityGML](image)
9.8.1 Land use object

LandUseType, LandUse

```xml
<xs:complexType name="LandUseType">
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="LandUseClassType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="LandUseUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

9.8.2 External code lists

The land use model introduces the following types, whose valid values are explicitly enumerated in an external code list (cf. chapter 7.5 and Annex A):

- LandUseClassType
- LandUseFunctionType
- LandUseUsageType
The grouping concept has already been introduced in chapter 7.7. *CityObjectGroups* are modelled using the so-called Composite Design Pattern from software engineering (cf. Gamma et al. 1995): *CityObjectGroups* aggregate *CityObjects* and furthermore are defined as special *CityObjects*. This implies that a group may become a member of another group realizing a recursive aggregation schema. However, in a CityGML instance document it has to be ensured (by the generating application) that no cyclic groupings are included. Fig. 42 shows the UML diagram for the class *CityObjectGroup*, for the XML schema see chapter 9.9.1.

The class *CityObjectGroup* has the optional attributes *class*, *function* and *usage*. In contrast to the other thematic classes, no code lists are defined for these attributes, because the reasons for groupings cannot be foreseen completely. The *class* attribute allows a group classification with respect to the stated function and may occur only once. The *function* attribute is intended to express the main purpose of a group, possibly to which thematic area it belongs (e.g. site, building, transportation, architecture, unknown etc.). The attribute *usage* can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times. Each member of a group may be qualified by a *role* name, reflecting the role each *CityObject* plays in the context of the group. Furthermore, a *CityObjectGroup* can optionally be assigned an arbitrary geometry object from the GML3 subset shown in Fig. 8 in chapter 8.1. This may be used to represent a generalized geometry generated from the members' geometries.

9.9.1 City object group

*CityObjectGroups*, *CityObjectGroup*
9.10 Generic objects and attributes

The concept of generic objects and attributes was introduced to ensure the storage and exchange of 3D objects, which are not covered by an explicitly modelled class within CityGML or which requires attributes not represented in CityGML. These generic extensions are realized by the classes `GenericCityObject` and `GenericAttribute`. Fig. 43 shows the UML diagram of generic objects and attributes, for XML schema definition see below and chapter 10.1.3.

![UML diagram of generic objects and attributes in CityGML.](image)

A `GenericCityObject` may have the attributes `class`, `function` and `usage` defined as `string`. The `class` attribute allows an object classification within the thematic area such as bridge, tunnel, pipe, power line, dam, or unknown. The `function` attribute describes to which thematic area the `GenericCityObject` belongs (e.g. site, transportation, architecture, energy supply, water supply, unknown etc.). The attribute `usage` can be used, if the way the object is actually used differs from the function. Both attributes can occur multiple times.

Every `_CityObject` can have an arbitrary number of `GenericAttributes`. Data types may be `String`, `Integer`, `Double` (floating point number), `URI` and `Date`. The attribute type is defined by the selection of the particular subclass (`StringAttribute`, `IntAttribute` etc.). `GenericAttributes` are inherited to all thematic subclasses of `CityObject`.

The geometry of a `GenericCityObject` can either be an explicit GML3 geometry or an `ImplicitGeometry` (see chapter 8.3). In the case of an explicit geometry the object can have only one geometry for each LOD, which may be an arbitrary 3D GML geometry object (class `_Geometry`, which is the base class of all GML geometries, `lodXGeometry`, `X in 0...4`). Absolute coordinates according to the reference system of the city model must be given for the explicit geometry. In the case of an `ImplicitGeometry`, a reference point (anchor point) of the object and optionally a transformation matrix must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates must be added. The shape of an `ImplicitGeometry` can be given as an external resource with a proprietary format, e.g. a VRML or DXF file from a local file system or an external web service. Alterna-
tively the shape can be specified as a 3D GML3 geometry with local cartesian coordinates using the property relativeGeometry (further details are given in chapter 8.3).

In order to specify the exact intersection of the DTM with the 3D geometry of a GenericCityObject, the latter can have TerrainIntersectionCurves for every LOD (cf. chapter 7.4). This is important for 3D visualization but also for certain applications like driving simulators. For example, if a bridge should be represented as a GenericCityObject, a smooth transition between the DTM and the road on the bridge would have to be ensured (in order to avoid unrealistic bumps).

9.10.1 Generic city object

GenericCityObjectType, GenericCityObject

9.10.2 Generic attributes

GenericTypeType, _genericAttribute, StringAttributeType, stringAttribute, etc.
<xs:element name="value" type="xs:integer" />
</xs:sequence>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="intAttribute" type="IntAttributeType" substitutionGroup="_genericAttribute" />
<!-- =========================================================================================  -->
<xs:complexType name="DoubleAttributeType">
<xs:complexContent>
<xs:extension base="_GenericAttributeType">
<xs:sequence>
<xs:element name="value" type="xs:double" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="doubleAttribute" type="DoubleAttributeType" substitutionGroup="_genericAttribute" />
<!-- =========================================================================================  -->
<xs:complexType name="DateAttributeType">
<xs:complexContent>
<xs:extension base="_GenericAttributeType">
<xs:sequence>
<xs:element name="value" type="xs:date" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="dateAttribute" type="DateAttributeType" substitutionGroup="_genericAttribute" />
<!-- =========================================================================================  -->
<xs:complexType name="UriAttributeType">
<xs:complexContent>
<xs:extension base="_GenericAttributeType">
<xs:sequence>
<xs:element name="value" type="xs:anyURI" />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="uriAttribute" type="UriAttributeType" substitutionGroup="_genericAttribute" />
9.11 Definition of code lists

For the representation of city object attributes having an enumerative range of values, the concept of dictionaries as provided by GML is used. The values of these attributes are defined in a file CityGML_ExternalCodeLists.xml, which comes with the CityGML schema document, but is not a normative part of this schema, since it may be modified, augmented, or replaced by other communities. The actual values in the file CityGML_ExternalCodeLists.xml are a suggestion of the SIG 3D.

The external code list file defines attribute values and assigns an unique identifier to each value. In a CityGML instance document, an attribute value is denoted by an identifier of a value, not by the value itself. Thus typos are avoided, and it is assured that the same concept is denoted the same way, by the same identifier and not by two different terms with identical meaning. Thus the use of code lists facilitates semantic and syntactic interoperability, since they define common terms within an information community. Furthermore, the dictionary concept enables to assign more than one term to the same dictionary entry, thus the same concept may be explained in different languages. To differentiate between the languages, code spaces are used.

An example for an enumerative attribute is RoofType, which is defined by the following excerpt of the external code list file:

```
<gml:DefinitionCollection gml:id="id228">
  <gml:name>RoofTypeType</gml:name>
  <gml:definitionMember>
    <gml:Definition gml:id="id229">
      <gml:name codeSpace="urn:sig3d:citygml:codelists">1000</gml:name>
      <gml:name>flat roof</gml:name>
    </gml:Definition>
    <gml:definitionMember>
      <gml:Definition gml:id="id230">
        <gml:name codeSpace="urn:sig3d:citygml:codelists">1010</gml:name>
        <gml:name>monopitch roof</gml:name>
      </gml:Definition>
    </gml:definitionMember>
  </gml:definitionMember>
</gml:DefinitionCollection>
```

In the dictionary concept, the values of an attribute are represented by a DefinitionCollection element, where each value is given by a Definition entry. In CityGML, a definition entry is identified by the name element, which is qualifies by the SIG 3D code space. The unqualified name element represents the value of the attribute. An optional description explains the value. CityGML does not use GML identifiers (gml:id) to link to attribute values, since IDs are restricted syntactically, and must be globally unique, which is not feasible for code lists.
10 XML schema definition (Normative)

10.1 Base Classes

10.1.1 Root element CityModel

```xml
<xs:schema targetNamespace="http://www.citygml.org/citygml/1/0/0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns="http://www.citygml.org/citygml/1/0/0" elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace="http://www.opengis.net/gml" schemaLocation="3.1.1/base/gml.xsd" />
<xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" schemaLocation="xAL/xAL.xsd"/>

<xs:complexType name="CityModelType">
  <xs:annotation>
    <xs:documentation>
      Type describing the "root" element of any city model file. It is a collection whose members are restricted to be features of a city model. All features are included as cityObjectMember.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureCollectionType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="CityModel" type="CityModelType" substitutionGroup="gml:_FeatureCollection"/>

10.1.2 Base class CityObject

```xml
<xs:complexType name="_CityObjectType" abstract="true">
  <xs:annotation>
    <xs:documentation>
      Type describing the abstract superclass of most CityGML features. Its purpose is to provide a creation and a termination date as well as a reference to corresponding objects in other information systems and generic attributes. A generalization relation may be used to relate features, which represent the same real-world object in different Levels-of-Detail, i.e. a feature and its generalized counterpart(s). The direction of this relation is from the feature to the corresponding generalized feature.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="creationDate" type="xs:date" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="terminationDate" type="xs:date" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="_CityObject" type="_CityObjectType" abstract="true" substitutionGroup="gml:_Feature"/>

10.1.3 Generic objects and attributes / generalization relation

```xml
<xs:complexType name="_GenericAttributeType" abstract="true">
  <xs:annotation>
    <xs:documentation>
      Generic (user defined) attributes may be used to represent attributes which are not covered explicitly by the CityGML schema. Generic attributes should be used with care; they should only be used if there is no appropriate attribute available in the schema. Otherwise, problems concerning semantic interoperability may arise. A generic attribute has a name and a value, which has further subclasses (IntAttribute, StringAttribute, ...).
    </xs:documentation>
  </xs:annotation>
  <xs:sequence />
</xs:complexType>
```
<xs:element name="_genericAttribute" type="_GenericAttributeType" abstract="true"/>
<xs:complexType name="StringAttributeType">
  <xs:annotation>
    <xs:documentation />
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_GenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="stringAttribute" type="StringAttributeType" substitutionGroup="_genericAttribute"/>

<xs:complexType name="IntAttributeType">
  <xs:annotation>
    <xs:documentation />
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_GenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:integer"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="intAttribute" type="IntAttributeType" substitutionGroup="_genericAttribute"/>

<xs:complexType name="DoubleAttributeType">
  <xs:annotation>
    <xs:documentation />
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_GenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:double"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="doubleAttribute" type="DoubleAttributeType" substitutionGroup="_genericAttribute"/>

<xs:complexType name="DateAttributeType">
  <xs:annotation>
    <xs:documentation />
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_GenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:date"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="dateAttribute" type="DateAttributeType" substitutionGroup="_genericAttribute"/>

<xs:complexType name="UriAttributeType">
  <xs:annotation>
    <xs:documentation />
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_GenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:anyURI"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="uriAttribute" type="UriAttributeType" substitutionGroup="_genericAttribute"/>
10.1.4 City object groups

A group may be used to aggregate arbitrary CityObjects according to some user-defined criteria. Examples for groups are the buildings in a specific region, the result of a query, or objects put together for visualization purposes. Each group has a name (inherited from AbstractGMLType), functions (e.g., building group), a class and zero or more usages. A geometry may optionally be attached to a group, if the geometry of the whole group differs from the geometry of the parts. Each member of a group may be qualified by a role name, reflecting the role each CityObject plays in the context of the group. As subclass of _CityObject, a CityObjectGroup inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. As CityObjectGroup itself is a CityObject, it may also contain groups. This type is defined as a CityObject as an XML complex-type.

<xs:complexType name="CityObjectGroupType">
  <xs:annotation>
    <xs:documentation>A group may be used to aggregate arbitrary CityObjects according to some user-defined criteria. Examples for groups are the buildings in a specific region, the result of a query, or objects put together for visualization purposes. Each group has a name (inherited from AbstractGMLType), functions (e.g., building group), a class and zero or more usages. A geometry may optionally be attached to a group, if the geometry of the whole group differs from the geometry of the parts. Each member of a group may be qualified by a role name, reflecting the role each CityObject plays in the context of the group. As subclass of _CityObject, a CityObjectGroup inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. As CityObjectGroup itself is a CityObject, it may also contain groups.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="xs:string" minOccurs="0" />
        <xs:element name="function" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="groupMember" type="CityObjectGroupMemberType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="geometry" type="gml:GeometryPropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
10.1.5 External references

Type describing the reference to a corresponding object in an other information system, for example in the german cadastral ALKIS, the german topographic information system or ATKIS, or the british 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, which contains an indication of the informationSystem.

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
10.2 Extensions to the GML geometry model

10.2.1 Special surfaces with material

The concept of positioning textures on surfaces complies with the standard X3D. Because there has been no appropriate texturing concept in GML3, CityGML adds the class TexturedSurface to the geometry model of GML 3. A texture is specified as a raster image referenced by an URI, and can be an arbitrary resource, even in the internet. Textures are positioned by employing the concept of texture coordinates, i.e. each texture coordinate matches with exactly one 3D coordinate of the TexturedSurface. The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry. Each surface may be assigned one or more appearances, each referring to one side of the surface.

```xml
<xs:complexType name="TexturedSurfaceType">
    <xs:annotation>
        <xs:documentation>
The concept of positioning textures on surfaces complies with the standard X3D. Because there has been no appropriate texturing concept in GML3, CityGML adds the class TexturedSurface to the geometry model of GML 3. A texture is specified as a raster image referenced by an URI, and can be an arbitrary resource, even in the internet. Textures are positioned by employing the concept of texture coordinates, i.e. each texture coordinate matches with exactly one 3D coordinate of the TexturedSurface. The use of texture coordinates allows an exact positioning and trimming of the texture on the surface geometry. Each surface may be assigned one or more appearances, each referring to one side of the surface.
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="gml:OrientableSurfaceType">
            <xs:sequence>
                <xs:element ref="appearance" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

---

A property that has an Appearance as its value domain, which can either be a Material (Color, ...) or a Texture. The Appearance Element can either be encapsulated in an element of this type or an XLink reference to a remote Appearance element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element must be given, but neither both nor none. The side of the surface the Appearance refers to is given by the orientation attribute, which refers to the corresponding sign attribute of the orientable surface: + means the side with positive orientation, and - the side with negative orientation.

```xml
<xs:complexType name="_AppearancePropertyType">
    <xs:annotation>
        <xs:documentation>
A property that has an Appearance as its value domain, which can either be a Material (Color, ...) or a Texture. The Appearance Element can either be encapsulated in an element of this type or an XLink reference to a remote Appearance element (where remote includes geometry elements located elsewhere in the same document). Either the reference or the contained element must be given, but neither both nor none. The side of the surface the Appearance refers to is given by the orientation attribute, which refers to the corresponding sign attribute of the orientable surface: + means the side with positive orientation, and - the side with negative orientation.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element ref="_Appearance" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="orientation" type="gml:SignType" default="+" />
    <xs:attributeGroup ref="gml:AssociationAttributeGroup" />
</xs:complexType>
```

---

This abstract type is the parent type of MaterialType and SimpleTextureType. It is derived from gml:AbstractGMLType, thus it inherits the attribute gml:id and may be referenced by an appearanceProperty, although it is defined elsewhere in another appearanceProperty.

```xml
<xs:complexType name="_AppearanceType" abstract="true">
    <xs:annotation>
        <xs:documentation>
This abstract type is the parent type of MaterialType and SimpleTextureType. It is derived from gml:AbstractGMLType, thus it inherits the attribute gml:id and may be referenced by an appearanceProperty, although it is defined elsewhere in another appearanceProperty.
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="_AppearanceType">
            <xs:sequence>
                <xs:element name="shininess" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="transparency" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="ambientIntensity" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="specularColor" type="Color" minOccurs="0" />
                <xs:element name="diffuseColor" type="Color" minOccurs="0" />
                <xs:element name="emissiveColor" type="Color" minOccurs="0" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

---

Adopted from X3D standard (http://www.web3d.org/x3d/)

```xml
<xs:complexType name="MaterialType">
    <xs:annotation>
        <xs:documentation>
Adopted from X3D standard (http://www.web3d.org/x3d/)
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="_AppearanceType">
            <xs:sequence>
                <xs:element name="shininess" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="transparency" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="ambientIntensity" type="doubleBetween0and1" minOccurs="0" />
                <xs:element name="specularColor" type="Color" minOccurs="0" />
                <xs:element name="diffuseColor" type="Color" minOccurs="0" />
                <xs:element name="emissiveColor" type="Color" minOccurs="0" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```
10.2.2 Implicit geometries

A property that has a Implicit Representation as its value domain, which is a representation of a geometry by referencing a prototype and transforming it to its real position in space.

Type for the implicit representation of a geometry. An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, e.g. a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate tuple of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. In order to determine the absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way even a projection might be modelled by the transformation matrix. The concept of implicit geometries is an enhancement of the geometry model of GML3.

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
<xs:complexType>
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType">
      <xs:sequence>
        <xs:element name="mimeType" type="MimeTypeType" minOccurs="0" />
        <xs:element name="transformationMatrix" type="TransformationMatrixType" minOccurs="0" />
        <xs:element name="libraryObject" type="xs:anyURI" minOccurs="0" />
        <xs:element name="relativeGMLGeometry" type="gml:GeometryPropertyType" minOccurs="0" />
        <xs:element name="referencePoint" type="gml:PointPropertyType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="ImplicitGeometry" type="ImplicitGeometryType" substitutionGroup="gml:_GML" />

<xs:simpleType name="MimeTypeType">
  <xs:annotation>
    <xs:documentation>
      MIME type of a geometry in an external library file. MIME types are defined by the IETF (Internet Engineering Task Force). The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>
10.3 Thematic Model

10.3.1 Sites

```xml
<xs:complexType name="_SiteType" abstract="true">
  <xs:annotation>
    <xs:documentation>
      Type describing the abstract superclass for buildings, facilities, etc. Future extensions of CityGML like bridges and tunnels would be modelled as subclasses of _Site. The german translation of site is 'Anlage'. As subclass of _CityObject, a _Site inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType" />
  </xs:complexContent>
</xs:complexType>
```

10.3.2 Buildings

```xml
<xs:element name="_Site" type="_SiteType" abstract="true" substitutionGroup="_CityObject" />
```
<xs:complexType name="BuildingPartType">
  <xs:complexContent>
    <xs:extension base="_AbstractBuildingType" />  
  </xs:complexContent>
</xs:complexType>

<xs:element name="BuildingPart" type="BuildingPartType" substitutionGroup="_AbstractBuilding" />

<xs:complexType name="BuildingType">
  <xs:complexContent>
    <xs:extension base="_AbstractBuildingType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Building" type="BuildingType" substitutionGroup="_AbstractBuilding" />

<xs:complexType name="BuildingPartPropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _AbstractBuilding to its building parts. The gml:AssociationType attribute group for enabling the use of refs is not repeated in the restriction and thus omitted. The building part has to be given inline, i.e. explicitly in this property. The reason for this inline definition is that no BuildingPart is used by more than one building, thus the use or references to building parts defined elsewhere is prohibited.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence>
        <xs:element ref="BuildingPart" />
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BuildingInstallationType">
  <xs:annotation>
    <xs:documentation>
      Intended function of a building. The values of this type are defined in the XML file "CityGML_ExternalCodeLists.xml", according to the dictionary concept of GML3. The values may be adopted from ALKIS, the german standard for cadastre modelling. If the cadastre models from other countries differ in the building functions, these values may be compiled in another codelist to be used with CityGML.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="xs:string" />  
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="RoofTypeType">
  <xs:annotation>
    <xs:documentation>
      Roof Types. The values of this type are defined in a XML file, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />  
</xs:complexType>

<xs:simpleType name="BuildingUsageType">
  <xs:annotation>
    <xs:documentation>
      Actual usage of a building. The values of this type are defined in a the XML file "CityGML_ExternalCodeLists.xml", according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />  
</xs:simpleType>

<xs:simpleType name="BuildingFunctionType">
  <xs:annotation>
    <xs:documentation>
      Intended function of a building. The values of this type are defined in the XML file "CityGML_ExternalCodeLists.xml", according to the dictionary concept of GML3. The values may be adopted from ALKIS, the german standard for cadastre modelling. If the cadastre models from other countries differ in the building functions, these values may be compiled in another codelist to be used with CityGML.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />  
</xs:simpleType>

<xs:simpleType name="BuildingClassType">
  <xs:annotation>
    <xs:documentation>
      Class of a building. The values of this type are defined in a the XML file "CityGML_ExternalCodeLists.xml", according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />  
</xs:simpleType>
A BuildingInstallation (German translation is 'Gebäudecharakteristik') is a part of a Building which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs. As subclass of _CityObject, a BuildingInstallation inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.

### BuildingInstallation

**Class of a building installation.** The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.

**Function of a building installation.** The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.

**Actual Usage of a building installation.** The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.

Denotes the relation of an AbstractBuilding to its building installations. The gml:AssociationType attribute group for enabling the use of refs is not repeated in the restriction and thus omitted. The building installation has to be given inline, i.e. explicitly in this property. The reason for this inline definition is that no installation of a building is used by more than one building, thus the use or references to building installations defined elsewhere is prohibited.

A BoundarySurface (German translation is 'Begrenzungsfläche') is a thematic object which classifies surfaces bounding a building or a room. The geometry of a BoundarySurface is given by MultiSurfaces. As it is a subclass of _CityObject, it inherits all attributes and relations, in particular the external references, the generic attributes, and the generalization relations.
<xs:element name="_BoundarySurface" type="_BoundarySurfaceType" abstract="true" substitutionGroup="_CityObject" />  
<xs:complexType name="RoofSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="RoofSurface" type="RoofSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="WallSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType">  
      <xs:sequence />  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="WallSurface" type="WallSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="GroundSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="GroundSurface" type="GroundSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="ClosureSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="ClosureSurface" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="FloorSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="FloorSurface" type="FloorSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="InteriorWallSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="CeilingSurfaceType">  
  <xs:complexContent>  
    <xs:extension base="_BoundarySurfaceType" />  
  </xs:complexContent>  
</xs:complexType>  
<xs:element name="CeilingSurface" type="CeilingSurfaceType" substitutionGroup="_BoundarySurface" />  
<xs:complexType name="BoundarySurfacePropertyType">  
  <xs:annotation>  
    <xs:documentation>Denotes the relation of an Building or Room to its bounding thematic surfaces (walls, roofs, ..). There is no differen- 
    tiation between interior surfaces bounding rooms and outer ones bounding buildings (one reason is, that ClosureSurfaces 
    belong to both types). It has to be made sure by additional integrity constraints that, e.g. a building is not related to CeilingSur-
    faces or a room not to RoofSurfaces.</xs:documentation>  
  </xs:annotation>  
  <xs:complexContent>  
    <xs:restriction base="gml:AssociationType">  
      <xs:sequence minOccurs="0">  
        <xs:element ref="_BoundarySurface" />  
      </xs:sequence>  
    </xs:restriction>  
  </xs:complexContent>  
</xs:complexType>

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
<xs:complexType name="OpeningPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an BoundarySurface to its openings (doors, windows). </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_Opening"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="_OpeningType" abstract="true">
  <xs:annotation>
    <xs:documentation>Type for openings (doors, windows) in walls. Used in LoD3 and LoD4 only. As subclass of _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="_Opening" type="_OpeningType" abstract="true" substitutionGroup="_CityObject"/>

<xs:complexType name="WindowType">
  <xs:annotation>
    <xs:documentation>Type for windows in walls. Used in LoD3 and LoD4 only. As subclass of _CityObject, a window inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_OpeningType">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Window" type="WindowType" substitutionGroup="_Opening"/>

<xs:complexType name="DoorType">
  <xs:annotation>
    <xs:documentation>Type for doors in walls. Used in LoD3 and LoD4 only. As subclass of _CityObject, a Door inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_OpeningType">
      <xs:sequence>
        <xs:element name="address" type="AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Door" type="DoorType" substitutionGroup="_Opening"/>

<xs:complexType name="RoomType">
  <xs:annotation>
    <xs:documentation>A Room is a thematic object for modelling the closed parts inside a building. It has to be closed, if necessary by using closure surfaces. The geometry may be either a solid, or a MultiSurface if the boundary is not topologically clean. The room connectivity may be derived by detecting shared thematic openings or closure surfaces: two rooms are connected if both use the same opening object or the same closure surface. The thematic surfaces bounding a room are referenced by the bound-
edBy property. As subclass of _CityObject, a Room inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. 

```xml
<xs:annotation>
  <xs:documentation>
    Class of a room. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3. 
  </xs:documentation>
</xs:annotation>
```

```xml
<xs:restriction base="xs:string" />
```

```xml
<xs:simpleType name="RoomFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of a room. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3. 
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>
```

```xml
<xs:simpleType name="RoomUsageType">
  <xs:annotation>
    <xs:documentation>
      Actual Usage of a room. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3. 
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>
```

```xml
<xs:complexType name="BuildingFurnitureType">
  <xs:annotation>
    <xs:documentation>
      Type for building furnitures. As subclass of _CityObject, a BuildingFurniture inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. 
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="BuildingFurnitureClassType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="function" type="BuildingFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="BuildingFurnitureUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod4ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```xml
<xs:element name="BuildingFurniture" type="BuildingFurnitureType" substitutionGroup="_CityObject" /> 
```

```xml
<xs:simpleType name="BuildingFurnitureClassType">
  <xs:annotation>
    <xs:documentation>
      Class of a building furniture. The values of this type are defined in a XML file, according to the dictionary concept of GML3. 
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>
```

```xml
<xs:simpleType name="BuildingFurnitureFunctionType">
  <xs:annotation>
    <xs:documentation>
      Class of a building furniture. The values of this type are defined in a XML file, according to the dictionary concept of GML3. 
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>
```
Function of a building furniture. The values of this type are defined in a XML file, according to the dictionary concept of GML3.

Actual Usage of a building Furniture. The values of this type are defined in a XML file, according to the dictionary concept of GML3.

Denotes the relation of an AbstractBuilding to its rooms. The gml:AssociationType attribute group for enabling the use of refs is not repeated in the restriction and thus omitted. The room has to be given inline within this property, not by reference.

Denotes the relation of a room to its interior furnitures (movable). The gml:AssociationType attribute group for enabling the use of refs is not repeated in the restriction and thus omitted. The BuildingFurniture has to be given inline within this property, not by reference.

Denotes the relation of an AbstractBuilding or a Door to its Addresses.

Denotes the relation of an AbstractBuilding to its Rooms (LoD4 only).

Denotes the relation of an AbstractBuilding or a Door to its Addresses.

Type for adresses. It references the xAL address standard issued by the OASIS consortium. Please note, that addresses are modelled as GML features. Every address can be assigned zero or more 2D or 3D point geometries (one gml:MultiPoint geometry) locating the entrance(s).
10.3.3 Transportation objects

<xsd:complexType name="_TransportationObject" type="_CityObjectType" substitutionGroup="_CityObject" />

<xsd:complexType name="TransportationComplexType">
  <xsd:annotation>
    <xsd:documentation>Type describing transportation complexes, which are aggregated features, e.g. roads, which consist of parts (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As subclass of _CityObject, a TransportationComplex inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="_TransportationObjectType">
      <xsd:sequence>
        <xsd:element name="function" type="TransportationComplexFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="usage" type="TransportationComplexUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="trafficArea" type="TrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="auxiliaryTrafficArea" type="AuxiliaryTrafficAreaPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="lod0Network" type="gml:GeometricComplexPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xsd:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xsd:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xsd:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="TransportationComplex" type="TransportationComplexType" substitutionGroup="_TransportationObject" />

<xsd:complexType name="TrafficAreaType">
  <xsd:annotation>
    <xsd:documentation>Type describing the class for traffic Areas. Traffic areas are the surfaces where traffic actually takes place. As subclass of _CityObject, a TrafficArea inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="_TransportationObjectType">
      <xsd:sequence>
        <xsd:element name="usage" type="TrafficAreaUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="function" type="TrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" minOccurs="0" />
        <xsd:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xsd:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xsd:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="TrafficArea" type="TrafficAreaType" substitutionGroup="_TransportationObject" />

<xs:complexType name="AuxiliaryTrafficAreaType">
  <xs:annotation>
    <xs:documentation>
    Type describing the class for auxiliary traffic Areas. These are the surfaces where no traffic actually takes place, but which belong to a transportation object. Examples are kerbstones, road markings and grass stripes. As subclass of _CityObject, an AuxiliaryTrafficArea inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_TransportationObjectType">
      <xs:sequence>
        <xs:element name="function" type="AuxiliaryTrafficAreaFunctionType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="surfaceMaterial" type="TrafficSurfaceMaterialType" minOccurs="0"/>
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="AuxiliaryTrafficArea" type="AuxiliaryTrafficAreaType" substitutionGroup="_TransportationObject"/>

<xs:complexType name="TrafficAreaPropertyType">
  <xs:annotation>
    <xs:documentation>
    Denotes the relation of a transportation complex to its parts, which are traffic areas in this case. Since an attribute group for enabling the use of references is provided, the relation may be given by a reference to another element defined elsewhere or by the complete TrafficArea inline.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="TrafficArea"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="AuxiliaryTrafficAreaPropertyType">
  <xs:annotation>
    <xs:documentation>
    Denotes the relation of a Transportation Complex to its parts, which are AuxiliaryTrafficAreas in this case. Since an attribute group for enabling the use of references is provided, the relation may be given by a reference to another AuxiliaryTrafficArea defined elsewhere or by the inline definition of an AuxiliaryTrafficArea.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="AuxiliaryTrafficArea"/>
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="TrackType">
  <xs:annotation>
    <xs:documentation>
    Type describing the class for tracks. A track is a small path mainly used by pedestrians. As subclass of _CityObject, a Track inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="TransportationComplexType"/>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Track" type="TrackType" substitutionGroup="TransportationComplex"/>

<xs:complexType name="RoadType">
  <xs:annotation>
    <xs:documentation>
    Type describing the class for roads. As subclass of _CityObject, a Road inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="TransportationComplexType"/>
  </xs:complexContent>
</xs:complexType>

<xs:element name="Road" type="RoadType"/>
<xs:extension base="TransportationComplexType">
  <xs:sequence />
</xs:extension>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="Road" type="RoadType" substitutionGroup="TransportationComplex" />
<!-- =========================================================================================  -->
<xs:complexType name="RailwayType">
  <xs:annotation>
    <xs:documentation>
      Type describing the class for railways. As subclass of _CityObject, a Railway inherits all attributes and relations,
      in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="Railway" type="RailwayType" substitutionGroup="TransportationComplex" />
<!-- =========================================================================================  -->
<xs:complexType name="SquareType">
  <xs:annotation>
    <xs:documentation>
      Type describing the class for squares. A square is an open area commonly found in cities (like a plaza). As sub-
      class of _CityObject, a Square inherits all attributes and relations, in particular an id, names, external references, generic at-
      tributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="TransportationComplexType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- =========================================================================================  -->
<xs:element name="Square" type="SquareType" substitutionGroup="TransportationComplex" />
<!-- =========================================================================================  -->
<xs:simpleType name="TransportationComplexFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of a transportation complex. The values of this type are defined in the XML file
      CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>

<!-- =========================================================================================  -->
<xs:simpleType name="TransportationComplexUsageType">
  <xs:annotation>
    <xs:documentation>
      Actual Usage of a transportation complex. The values of this type are defined in the XML file
      CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>

<!-- =========================================================================================  -->
<xs:simpleType name="TrafficAreaFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of a traffic area. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>

<!-- =========================================================================================  -->
<xs:simpleType name="AuxiliaryTrafficAreaFunctionType">
  <xs:annotation>
    <xs:documentation>
      Function of an auxiliary traffic area. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>

<!-- =========================================================================================  -->
<xs:simpleType name="TrafficAreaUsageType">
  <xs:annotation>
    <xs:documentation>
      Usage of a traffic area. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string" />
</xs:simpleType>

88
10.3.4 Vegetation objects

<!-- =========================================================================================  -->
<xs:complexType name="_VegetationObjectType" abstract="true">
    <xs:annotation>
        <xs:documentation>
            Type describing the abstract superclass for vegetation objects. A subclass is either a SolitaryVegetationObject or 
a PlantCover. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="_CityObjectType" />
    </xs:complexContent>
</xs:complexType>
<!-- =========================================================================================  -->
<xs:element name="_VegetationObject" type="_VegetationObjectType" substitutionGroup="_CityObject" />  
<!-- =========================================================================================  -->
<xs:complexType name="PlantCoverType">
    <xs:annotation>
        <xs:documentation>
            Type describing Plant Covers resp. Biotopes (German translation: Vegetation). As subclass of _CityObject, a 
VegetationObject inherits all attributes and relations, in particular an id, names, external references, generic attributes and gen-
eralization relations. </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="_VegetationObjectType">
            <xs:sequence>
                <xs:element name="class" type="PlantCoverClassType" minOccurs="0" maxOccurs="1" />
                <xs:element name="function" type="PlantCoverFunctionType" minOccurs="0" maxOccurs="0" />
                <xs:element name="averageHeight" type="gml:LengthType" minOccurs="0" />
                <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
                <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
                <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
                <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
                <xs:element name="lod1MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
                <xs:element name="lod2MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
                <xs:element name="lod3MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
                <xs:element name="lod4MultiSolid" type="gml:MultiSolidPropertyType" minOccurs="0" />
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- =========================================================================================  -->
<xs:element name="PlantCover" type="PlantCoverType" substitutionGroup="_VegetationObject" />  
<!-- =========================================================================================  -->
<xs:simpleType name="PlantCoverClassType">
    <xs:annotation>
        <xs:documentation>
            Class of a PlantCover. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, ac-
cording to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string" />
</xs:simpleType>
<!-- =========================================================================================  -->
<xs:simpleType name="PlantCoverFunctionType">
    <xs:annotation>
        <xs:documentation>
            Function of a PlantCover. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, ac-
cording to the dictionary concept of GML3. </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string" />
</xs:simpleType>
<!-- =========================================================================================  -->
<xs:complexType name="SolitaryVegetationObjectType">
    <xs:annotation>
        <xs:documentation>
            Type describing solitary vegetation objects, e.g., trees. Its geometry is either defined explicitly by a GML 3 ge-
ometry with absolute coordinates, or in the case of multiple occurrences of the same vegetation object, implicitly by a reference 
to a shape definition and a transformation. The shape definition may be given in an external file. As subclass of _CityObject, a 
SolitaryVegetationObject inherits all attributes and relations, in particular an id, names, external references, generic attributes 
and generalization relations. </xs:documentation>
    </xs:annotation>
</xs:complexType>
<!-- =========================================================================================  -->
<xs:element name="SolitaryVegetationObject" type="SolitaryVegetationObjectType" substitutionGroup="_CityObject" />

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
10.3.5 Water bodies

<x:simpleType name="WaterBodyType">
   <xs:annotation>
      <xs:documentation>Type describing Water Bodies, e.g., lakes, rivers. As subclass of _CityObject, a WaterBody inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
   </xs:annotation>
</xs:simpleType>
the water body is influenced by the tide. The values of this type are defined in the XML file CityGML_ExternalCodeLists.xml, according to the dictionary concept of GML3.

````xml
<xs:complexType name="WaterSurfaceType">
  <xs:annotation>
    <xs:documentation>
      Type describing the surface of a water body, which separates the water from the air. As subclass of _CityObject, a WaterSurface inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_WaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element name="waterLevel" type="WaterLevelType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
````

````xml
<xs:element name="WaterSurface" type="WaterSurfaceType" substitutionGroup="_WaterBoundarySurface" />
````

````xml
<xs:complexType name="WaterGroundSurfaceType">
  <xs:annotation>
    <xs:documentation>
      Type describing the ground surface of a water body, i.e. the boundary to the digital terrain model. As subclass of _CityObject, a WaterGroundSurface inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_WaterBoundarySurfaceType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
````

````xml
<xs:element name="WaterGroundSurface" type="WaterGroundSurfaceType" substitutionGroup="_WaterBoundarySurface" />
````

````xml
<xs:complexType name="WaterClosureSurfaceType">
  <xs:annotation>
    <xs:documentation>
      Type describing the closure surface between water bodys. As subclass of _CityObject, a WaterClosureSurface inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_WaterBoundarySurfaceType">
      <xs:sequence />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
````

````xml
<xs:element name="WaterClosureSurface" type="WaterClosureSurfaceType" substitutionGroup="_WaterBoundarySurface" />
````

10.3.6  City furniture

````xml
<xs:complexType name="CityFurnitureType">
  <xs:annotation>
    <xs:documentation>
      Type describing city furnitures, like traffic lights, benches, ... As subclass of _CityObject, a CityFurniture inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="CityFurnitureClassType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="function" type="CityFurnitureFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod1Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod1ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod2ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod3ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod4ImplicitRepresentation" type="ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
````
10.3.7 Land use

<xs:complexType name="LandUseType">
  <xs:annotation>
    <xs:documentation>Type describing the class for Land Use in all LoD. LandUse objects describe areas of the earth’s surface dedicated to a specific land use. The geometry must consist of 3-D surfaces. As subclass of _CityObject, a LandUse inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="class" type="LandUseClassType" minOccurs="0" />
        <xs:element name="function" type="LandUseFunctionType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="usage" type="LandUseUsageType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
        <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
10.3.8  Digital Terrain Model

-xs:complexType name="ReliefFeatureType">
  -xs:annotation>
    -xs:documentation>Type describing the features of the Digital Terrain Model. As subclass of _CityObject, a ReliefFeature inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="lod" type="integerBetween0and4" />
        <xs:element name="reliefComponent" type="ReliefComponentPropertyType" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

-xs:complexType name="_ReliefComponentType" abstract="true">
  -xs:annotation>
    -xs:documentation>Type describing the components of a relief feature - either a TIN, a Grid, mass points or break lines. As subclass of _CityObject, a ReliefComponent inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_CityObjectType">
      <xs:sequence>
        <xs:element name="lod" type="integerBetween0and4" />
        <xs:element name="extent" type="gml:PolygonPropertyType" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

-xs:complexType name="ReliefComponentPropertyType">
  -xs:annotation>
    -xs:documentation>Denotes the relation of a relief feature to its components. The relation may be given by a reference to a component defined elsewhere or by the complete inline definition of a component.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence minOccurs="0">
        <xs:element ref="_ReliefComponent" />
      </xs:sequence>
      <xs:attributeGroup ref="gml:AssociationAttributeGroup" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

-xs:complexType name="TINReliefType">
  -xs:annotation>
    -xs:documentation>Type describing the TIN component of a relief feature. As subclass of _CityObject, a TINRelief inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="_ReliefComponentType">
      <xs:sequence>
        <xs:element name="tin" type="tinPropertyType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

-xs:complexType name="RasterReliefType">
  -xs:annotation>
    -xs:documentation>Type describing the raster component of a relief feature. As subclass of _CityObject, a RasterRelief inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.</xs:documentation>
  </xs:annotation>
10.4 Definition of restricted types
Type for values, which are greater or equal than 0 and less or equal than 1. Used for color encoding, for example.

```
<xs:simpleType name="doubleBetween0and1List">
  <xs:annotation>
    <xs:documentation>List for double values, which are greater or equal than 0 and less or equal than 1. Used for color encoding, for example.
    </xs:documentation>
  </xs:annotation>
  <xs:list itemType="doubleBetween0and1"/>
</xs:simpleType>
```

```
<xs:simpleType name="TransformationMatrixType">
  <xs:annotation>
    <xs:documentation>Used for implicit geometries. The Transformation matrix is a 4 by 4 matrix, thus it must be a list with 16 items. The order the matrix element are represented is row-major, i.e. the first 4 elements represent the first row, the fifth to the eight element the second row,...</xs:documentation>
  </xs:annotation>
  <xs:restriction base="gml:doubleList">
    <xs:minLength value="16"/>
    <xs:maxLength value="16"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="integerBetween0and4">
  <xs:annotation>
    <xs:documentation>Type for integer values, which are greater or equal than 0 and less or equal than 4. Used for encoding of the LoD number.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="4"/>
  </xs:restriction>
</xs:simpleType>
```
11 Annex

11.1 Annex A: External code lists

For most of the external code lists of CityGML the corresponding values are given in this chapter. The following values are a proposal of the SIG 3D and may be extended or replaced by other communities to fit their needs. The external code list for roof types is given in XML format below, while the others are depicted in tabular form for space reasons.

External code list for roof types:

```xml
<?xml version='1.0' encoding='UTF-8'?>
<gml:Dictionary gml:id="InternalCodeLists"
xmlns="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gml ../3.1.1/base/gml.xsd">
<gml:description>Internal Code Lists for CityGML as proposed by the SIG 3D of GDI NRW</gml:description>
<gml:name>CityGML</gml:name>
<gml:dictionaryEntry>
<gml:DefinitionCollection gml:id="id228">
<gml:description></gml:description>
<gml:name>RoofTypeType</gml:name>
<gml:definitionMember>
<gml:Definition gml:id="id229">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1000</gml:name>
<gml:name>flat roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id230">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1010</gml:name>
<gml:name>monopitch roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id231">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1020</gml:name>
<gml:name>skip pent roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id232">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1030</gml:name>
<gml:name>gabled roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id233">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1040</gml:name>
<gml:name>hipped roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id234">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1050</gml:name>
<gml:name>half-hipped roof</gml:name>
</gml:Definition>
</gml:definitionMember>
<gml:definitionMember>
<gml:Definition gml:id="id235">
<gml:description></gml:description>
<gml:name codeSpace="urn:sig3d:citygml:codelists">1060</gml:name>
<gml:name>mansard roof</gml:name>
</gml:Definition>
</gml:definitionMember>
</gml:DefinitionCollection>
</gml:dictionaryEntry>
</gml:Dictionary>
```

Copyright © 2006 Open Geospatial Consortium, Inc. All Rights Reserved.
The following code lists for the attributes BuildingFunctionType, CityFurnitureFunctionType, CityFurnitureClassType, TrafficAreaUsageType and TrafficAreaFunctionType are based on the German cadastre standard ALKIS or on the German official standard for topographic modelling ATKIS (http://www.atkis.de/).

<table>
<thead>
<tr>
<th>BuildingFunctionType</th>
<th>1000</th>
<th>residential building</th>
<th>1780</th>
<th>heat plant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>tenement</td>
<td>1790</td>
<td>pumping station</td>
<td></td>
</tr>
<tr>
<td>1020</td>
<td>hostel</td>
<td>1800</td>
<td>building for disposal</td>
<td></td>
</tr>
<tr>
<td>1030</td>
<td>residential- and administration building</td>
<td>1810</td>
<td>building for effluent disposal</td>
<td></td>
</tr>
<tr>
<td>1040</td>
<td>residential- and office building</td>
<td>1820</td>
<td>building for filter plant</td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>residential- and business building</td>
<td>1830</td>
<td>toilet</td>
<td></td>
</tr>
<tr>
<td>1060</td>
<td>residential- and plant building</td>
<td>1840</td>
<td>rubbish bunker</td>
<td></td>
</tr>
<tr>
<td>1070</td>
<td>agrarian- and forestry building</td>
<td>1850</td>
<td>building for garbage incineration</td>
<td></td>
</tr>
<tr>
<td>1080</td>
<td>residential- and commercial building</td>
<td>1860</td>
<td>building for abatement disposal</td>
<td></td>
</tr>
<tr>
<td>1090</td>
<td>forester's house</td>
<td>1870</td>
<td>building for agrarian and forestry</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Year</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------------------------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>holiday house</td>
<td>1880</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td>summer house</td>
<td>1890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1120</td>
<td>office building</td>
<td>1900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td>credit institution</td>
<td>1910</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1140</td>
<td>affirmation</td>
<td>1920</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1150</td>
<td>business building</td>
<td>1930</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1160</td>
<td>store</td>
<td>1940</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1170</td>
<td>shopping centre</td>
<td>1950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1180</td>
<td>kiosk</td>
<td>1960</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1190</td>
<td>apothecary</td>
<td>1970</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td>pavilion</td>
<td>1980</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1210</td>
<td>hotel</td>
<td>1990</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1220</td>
<td>youth hostel</td>
<td>2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1230</td>
<td>campground building</td>
<td>2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1240</td>
<td>restaurant</td>
<td>2020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1250</td>
<td>cafeteria</td>
<td>2030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1260</td>
<td>recreational site</td>
<td>2040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1270</td>
<td>festival room</td>
<td>2050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1280</td>
<td>cinema</td>
<td>2060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1290</td>
<td>bowling alley</td>
<td>2070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>casino</td>
<td>2080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1310</td>
<td>industrial building</td>
<td>2090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1320</td>
<td>factory</td>
<td>2100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1330</td>
<td>repair shop</td>
<td>2110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1340</td>
<td>gas station</td>
<td>2120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1350</td>
<td>washing plant</td>
<td>2130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1360</td>
<td>cold store</td>
<td>2140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1370</td>
<td>depot</td>
<td>2150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1380</td>
<td>building for research purposes</td>
<td>2160</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1390</td>
<td>pit</td>
<td>2170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td>saline</td>
<td>2180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1410</td>
<td>miscellaneous building for industry</td>
<td>2190</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1420</td>
<td>mill</td>
<td>2200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1430</td>
<td>windmill</td>
<td>2210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1440</td>
<td>water mill</td>
<td>2220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1450</td>
<td>bucket elevator</td>
<td>2230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1460</td>
<td>meteorological station</td>
<td>2240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1470</td>
<td>plant at traffic centre</td>
<td>2250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1480</td>
<td>maintenance area</td>
<td>2260</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1490</td>
<td>concourse</td>
<td>2270</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>signalman station</td>
<td>2280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1510</td>
<td>locomotive hall</td>
<td>2290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1520</td>
<td>positioner</td>
<td>2300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1530</td>
<td>plant building for air traffic</td>
<td>2310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1540</td>
<td>hangar</td>
<td>2320</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1550</td>
<td>plant for shipping traffic</td>
<td>2330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1560</td>
<td>shipyard</td>
<td>2340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1570</td>
<td>dock</td>
<td>2350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1580</td>
<td>plant for lock</td>
<td>2360</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1590</td>
<td>boathouse</td>
<td>2370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td>plant for cableway</td>
<td>2380</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1610</td>
<td>parking block</td>
<td>2390</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1620</td>
<td>parking deck</td>
<td>2400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Building/Installation/Function Type</td>
<td>Code</td>
<td>City/Furniture/Function Type</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------</td>
<td>------</td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>1630</td>
<td>garage</td>
<td>2410</td>
<td>fire department</td>
<td></td>
</tr>
<tr>
<td>1640</td>
<td>vehicle hall</td>
<td>2420</td>
<td>barracks</td>
<td></td>
</tr>
<tr>
<td>1650</td>
<td>underground garage</td>
<td>2430</td>
<td>bunker</td>
<td></td>
</tr>
<tr>
<td>1660</td>
<td>building for accommodation</td>
<td>2440</td>
<td>penitentiary</td>
<td></td>
</tr>
<tr>
<td>1670</td>
<td>waterworks</td>
<td>2450</td>
<td>cemetery building</td>
<td></td>
</tr>
<tr>
<td>1680</td>
<td>pump station</td>
<td>2460</td>
<td>funerual hall</td>
<td></td>
</tr>
<tr>
<td>1690</td>
<td>basin</td>
<td>2470</td>
<td>crematorium</td>
<td></td>
</tr>
<tr>
<td>1700</td>
<td>electric power station</td>
<td>2480</td>
<td>railroad station</td>
<td></td>
</tr>
<tr>
<td>1710</td>
<td>transformer station</td>
<td>2490</td>
<td>airport building</td>
<td></td>
</tr>
<tr>
<td>1720</td>
<td>converter</td>
<td>2500</td>
<td>building for underground station</td>
<td></td>
</tr>
<tr>
<td>1730</td>
<td>dry well</td>
<td>2510</td>
<td>building for tramway</td>
<td></td>
</tr>
<tr>
<td>1740</td>
<td>turbine house</td>
<td>2520</td>
<td>building for bus station</td>
<td></td>
</tr>
<tr>
<td>1750</td>
<td>boiler house</td>
<td>2530</td>
<td>acceptance building navigation</td>
<td></td>
</tr>
<tr>
<td>1760</td>
<td>building for communications</td>
<td>2540</td>
<td>building for recreation purposes</td>
<td></td>
</tr>
<tr>
<td>1770</td>
<td>gas works</td>
<td>9999</td>
<td>unknown</td>
<td></td>
</tr>
</tbody>
</table>

**Building/Installation/Function Type**

- Balcony
- Winter garden
- Arcade
- Chimney at the building

**City/Furniture/Function Type**

- Traffic
- Communication

**City/Furniture/Class Type**

- Communication adjustment
- Telephone house
- Postbox
- Emergency call adjustment
- Fire detector
- Silent alarm column
- Switching unit
- Road sign
- Traffic light
- Free-standing sign
- Free-standing warning sign
- Bus stop
- Milestone
- Railroad crossing
- Gate
- Latern
- Column
- Stacking lamp
- Flagpole
- Road sinking box
- Rubbish box
- Clock
- Leveling head light
- Floodlight mast
- Windmill
- Solar cell
- Water wheel
### SurfaceMaterialType

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>asphalt</td>
<td>8</td>
<td>soil</td>
</tr>
<tr>
<td>2</td>
<td>concrete</td>
<td>9</td>
<td>sand</td>
</tr>
<tr>
<td>3</td>
<td>pavement</td>
<td>10</td>
<td>grass</td>
</tr>
<tr>
<td>4</td>
<td>cobblestone</td>
<td>11</td>
<td>wood</td>
</tr>
<tr>
<td>5</td>
<td>gravel</td>
<td>12</td>
<td>steel</td>
</tr>
<tr>
<td>6</td>
<td>rail_with_bed</td>
<td>13</td>
<td>marble</td>
</tr>
<tr>
<td>7</td>
<td>rail_without_bed</td>
<td>99990</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### TrafficAreaUsageType

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pedestrian</td>
<td>9</td>
<td>train</td>
</tr>
<tr>
<td>2</td>
<td>car</td>
<td>10</td>
<td>boat, ferry, ship</td>
</tr>
<tr>
<td>3</td>
<td>truck</td>
<td>11</td>
<td>teleferic</td>
</tr>
<tr>
<td>4</td>
<td>bus, taxi</td>
<td>12</td>
<td>aeroplane</td>
</tr>
<tr>
<td>5</td>
<td>train</td>
<td>13</td>
<td>helicopter</td>
</tr>
<tr>
<td>6</td>
<td>cyclist</td>
<td>14</td>
<td>taxi</td>
</tr>
<tr>
<td>7</td>
<td>motorcyclist</td>
<td>15</td>
<td>horse</td>
</tr>
<tr>
<td>8</td>
<td>tram, streetcar</td>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### TrafficAreaFunctionType

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>driving_lane</td>
<td>21</td>
<td>crosswalk</td>
</tr>
<tr>
<td>2</td>
<td>footpath</td>
<td>22</td>
<td>barrier</td>
</tr>
<tr>
<td>3</td>
<td>cyclepath</td>
<td>23</td>
<td>stairs</td>
</tr>
<tr>
<td>4</td>
<td>combined foot-/cyclepath</td>
<td>24</td>
<td>escalator</td>
</tr>
<tr>
<td>5</td>
<td>square</td>
<td>25</td>
<td>filtering lane</td>
</tr>
<tr>
<td>6</td>
<td>car_park</td>
<td>26</td>
<td>airport_runway</td>
</tr>
<tr>
<td>7</td>
<td>parking_lay_by</td>
<td>27</td>
<td>airport_taxiway</td>
</tr>
<tr>
<td>8</td>
<td>rail</td>
<td>28</td>
<td>airport_apron</td>
</tr>
<tr>
<td>9</td>
<td>rail_road_combined</td>
<td>29</td>
<td>airport_heliport</td>
</tr>
<tr>
<td>10</td>
<td>drainage</td>
<td>30</td>
<td>airport_runway_marking</td>
</tr>
<tr>
<td>11</td>
<td>curbstone</td>
<td>31</td>
<td>green spaces</td>
</tr>
<tr>
<td>12</td>
<td>road marking</td>
<td>32</td>
<td>flower tubs</td>
</tr>
<tr>
<td>13</td>
<td>road_marking_direction</td>
<td>33</td>
<td>recreation</td>
</tr>
<tr>
<td>14</td>
<td>road_marking_lane</td>
<td>34</td>
<td>bus_lay_by</td>
</tr>
<tr>
<td>15</td>
<td>road_marking_restricted</td>
<td>35</td>
<td>motorway</td>
</tr>
<tr>
<td>16</td>
<td>road_marking_crosswalk</td>
<td>36</td>
<td>motorway_entry</td>
</tr>
<tr>
<td>17</td>
<td>road_marking_stop</td>
<td>37</td>
<td>motorway_exit</td>
</tr>
<tr>
<td>18</td>
<td>road_marking_other</td>
<td>38</td>
<td>motorway_emergency lane</td>
</tr>
<tr>
<td>19</td>
<td>overhead wire (trolley)</td>
<td>39</td>
<td>private_area</td>
</tr>
<tr>
<td>20</td>
<td>train platform</td>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>
The following three external code lists for vegetation objects should be considered as discussion proposals. The tables show a short excerpt from the complete lists. Please note, that further discussion is needed to get code lists for vegetation objects which have clear semantics. The following table is based on http://www.biologie.uni-hamburg.de/b-online/e57/57.htm (Tab.3)

### PlantCoverClassType (excerpt)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>Lemnetea</td>
<td>1280</td>
<td>Arrhenatheretea</td>
</tr>
<tr>
<td>1020</td>
<td>Asplenietea rupestris</td>
<td>1290</td>
<td>Molinio-Juncetea</td>
</tr>
<tr>
<td>1030</td>
<td>Adiantetea</td>
<td>1300</td>
<td>Scheuchzerio-Caricetea fuscae azi-dophile</td>
</tr>
<tr>
<td>1040</td>
<td>Thlaspietea rotundifolii</td>
<td>1310</td>
<td>Festuco-Brometea</td>
</tr>
<tr>
<td>1050</td>
<td>Crithmo-Limonietea</td>
<td>1320</td>
<td>Elyno- Seslerietea</td>
</tr>
<tr>
<td>1060</td>
<td>Ammophietea</td>
<td>1330</td>
<td>Caricetea curvulae azidophile</td>
</tr>
<tr>
<td>1070</td>
<td>Cakiletea maritimae halophile</td>
<td>1340</td>
<td>Calluno-Ulicetea</td>
</tr>
<tr>
<td>1080</td>
<td>Secalinetea</td>
<td>1350</td>
<td>Oxyccoco-Sphagnetea</td>
</tr>
<tr>
<td>1090</td>
<td>Chenopodietea</td>
<td>1360</td>
<td>Salicetea purpureae</td>
</tr>
<tr>
<td>1100</td>
<td>Onopordetea</td>
<td>1370</td>
<td>Betulo-Adenostylete</td>
</tr>
<tr>
<td>1110</td>
<td>Epilobietea angustifolii</td>
<td>1380</td>
<td>Alnetea glutinosae</td>
</tr>
<tr>
<td>1120</td>
<td>Bidentetea tripartiti</td>
<td>1390</td>
<td>Erico-Pinetea</td>
</tr>
<tr>
<td>1130</td>
<td>Zoosteretea marinae halophile</td>
<td>1400</td>
<td>Vaccinio-Piceetea</td>
</tr>
<tr>
<td>1140</td>
<td>Ruppietea maritimae</td>
<td>1410</td>
<td>Quercetea robor-petraeae</td>
</tr>
<tr>
<td>1150</td>
<td>Potametap haftend</td>
<td>1420</td>
<td>Querco-Fagetea</td>
</tr>
<tr>
<td>1160</td>
<td>Litoreletea</td>
<td>1430</td>
<td>Crithmo-Staticetea</td>
</tr>
<tr>
<td>1170</td>
<td>Plantaginetea majoris</td>
<td>1440</td>
<td>Tuberarieteca gutti</td>
</tr>
<tr>
<td>1180</td>
<td>Isoeto-Nanojuncetea</td>
<td>1450</td>
<td>Juncetea maritimae</td>
</tr>
<tr>
<td>1190</td>
<td>Montino-Cardaminetee</td>
<td>1460</td>
<td>Thero-Brachypodietea</td>
</tr>
<tr>
<td>1200</td>
<td>Corynephoretrea</td>
<td>1470</td>
<td>Ononido-Rosmarinetea</td>
</tr>
<tr>
<td>1210</td>
<td>Astereteea tripolium</td>
<td>1480</td>
<td>Nerio-Tamaricetea</td>
</tr>
<tr>
<td>1220</td>
<td>Salicorinetea</td>
<td>1490</td>
<td>Pegano-Salsoletea</td>
</tr>
<tr>
<td>1230</td>
<td>Juncetea maritimi</td>
<td>1500</td>
<td>Cisto-Lavanduletea</td>
</tr>
<tr>
<td>1240</td>
<td>Phragmiteteea</td>
<td>1510</td>
<td>Quercetea Ilicis</td>
</tr>
<tr>
<td>1250</td>
<td>Spartinetea</td>
<td>1520</td>
<td>Populetea albae</td>
</tr>
<tr>
<td>1260</td>
<td>Sedo-Scleranthetee</td>
<td>9999</td>
<td>unknown</td>
</tr>
<tr>
<td>1270</td>
<td>Saliceteea herbaceae</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following two code lists are based on information extracted from http://www.bundessortenamt.de and http://www.forst-hamburg.de/baumarten.htm

### SpeciesType (excerpt)

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1640</td>
<td>Abies alba</td>
<td>1790</td>
</tr>
<tr>
<td>1650</td>
<td>Abies cephalonica</td>
<td>1800</td>
</tr>
<tr>
<td>1660</td>
<td>Abies concolor</td>
<td>1810</td>
</tr>
<tr>
<td>1670</td>
<td>Abies grandis</td>
<td>1820</td>
</tr>
<tr>
<td>1680</td>
<td>Abies homolepsis</td>
<td>1830</td>
</tr>
<tr>
<td>1690</td>
<td>Abies koreana</td>
<td>1840</td>
</tr>
<tr>
<td>1700</td>
<td>Abies lasiocarpa</td>
<td>1850</td>
</tr>
<tr>
<td>1710</td>
<td>Abies nordmanniana</td>
<td>1860</td>
</tr>
<tr>
<td>1720</td>
<td>Abies pinsapo</td>
<td>1870</td>
</tr>
<tr>
<td>1730</td>
<td>Abies procerav</td>
<td>1880</td>
</tr>
<tr>
<td>1740</td>
<td>Abies procerav ‘Glauc’</td>
<td>1890</td>
</tr>
<tr>
<td>1750</td>
<td>Abies veitchii</td>
<td>1900</td>
</tr>
<tr>
<td>1760</td>
<td>Acer campéstre</td>
<td>1910</td>
</tr>
<tr>
<td>Number</td>
<td>Species</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>1770</td>
<td>Acer capillipes</td>
<td></td>
</tr>
<tr>
<td>1780</td>
<td>Acer cappadocicum</td>
<td></td>
</tr>
<tr>
<td>1720</td>
<td>Acer tartaricum</td>
<td></td>
</tr>
</tbody>
</table>

**PlantClassType**

<table>
<thead>
<tr>
<th>Class Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shrub</td>
<td>1000</td>
<td>base plants</td>
</tr>
<tr>
<td>conifer</td>
<td>1060</td>
<td>medium high plants</td>
</tr>
<tr>
<td>deciduous tree</td>
<td>1070</td>
<td>high plants</td>
</tr>
<tr>
<td>bushes</td>
<td>1080</td>
<td>grasses</td>
</tr>
<tr>
<td>aquatic plants</td>
<td>1090</td>
<td>ferns</td>
</tr>
<tr>
<td>climber</td>
<td>1100</td>
<td>unknown</td>
</tr>
</tbody>
</table>

The MIME types given in the following table are defined by the Internet Assigned Numbers Authority (IANA), see http://www.iana.org/. Generally, the MIME format is standardized by the Internet Engineering Task Force (IETF), see http://www.ietf.org/. Unlike the other code lists the MIME types are not represented by numbers, but instead use their given identifier.

**MimeTypeType**

<table>
<thead>
<tr>
<th>MIME Type</th>
<th>VRML97</th>
<th>Shockwave 3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>model/vrml</td>
<td>VRML97</td>
<td>model/x3d+vrml</td>
</tr>
<tr>
<td>application/x-3ds</td>
<td>3ds max</td>
<td>model/x3d+binary</td>
</tr>
<tr>
<td>application/dxf</td>
<td>AutoCad DXF</td>
<td>model/x3d+xml</td>
</tr>
<tr>
<td>application/x-autocad</td>
<td>AutoCad DXF</td>
<td>X3D</td>
</tr>
<tr>
<td>application/x-dxf</td>
<td>AutoCad DXF</td>
<td>X3D</td>
</tr>
<tr>
<td>application/acad</td>
<td>AutoCad DWG</td>
<td>X3D</td>
</tr>
</tbody>
</table>
11.2  *Annex B: Overview of employed GML3 geometry classes*

<table>
<thead>
<tr>
<th>Abstract GML classes referenced in CityGML</th>
<th>GML subclass actually used in CityGML</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Geometry</td>
<td></td>
</tr>
<tr>
<td>_Solid</td>
<td>Solid (boundary is restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td>CompositeSolid</td>
<td></td>
</tr>
<tr>
<td>_Surface</td>
<td>Polygon (with holes, modeled by Rings. The boundary is restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td>OrientableSurface (base surface is restricted to a Polygon)</td>
</tr>
<tr>
<td></td>
<td>TexturedSurface (defined in CityGML, not in GML. For restrictions see OrientableSurface)</td>
</tr>
<tr>
<td></td>
<td>CompositeSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td></td>
<td>TriangulatedSurface</td>
</tr>
<tr>
<td></td>
<td>Tin</td>
</tr>
<tr>
<td>Curve</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LineString</td>
</tr>
<tr>
<td></td>
<td>CompositeCurve (members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Point</td>
</tr>
<tr>
<td>Grid</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RectifiedGridCoverage</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MultiSolid</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MultiSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td></td>
<td>MultiCurve(members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td></td>
<td>MultiPoint</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeometricComplex (restricted to connected, linear networks)</td>
</tr>
</tbody>
</table>

11.3  *Annex C: Overview of the assignment of features to LODs*

The following table lists all feature types of CityGML. For each type, all non-spatial and spatial properties are given, including its type. Each feature is assigned a range of LOD in which it may occur, and for each spatial property the LOD in which it represents the feature is stated.

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Property</th>
<th>Type</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityModelType</td>
<td>cityObjectMember</td>
<td>gml:FeaturePropertyType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>CityObjectType</td>
<td>creationDate</td>
<td>xs:date</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>terminationDate</td>
<td>xs:date</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>externalReference</td>
<td>ExternalReferenceType</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>genericAttribute</td>
<td>GenericAttributeType</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>generalizesTo</td>
<td>GeneralizationRelationType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>GenericCityObjectType</td>
<td>function</td>
<td>xs:string</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>xs:string</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>xs:string</td>
<td>0 – 4</td>
</tr>
<tr>
<td></td>
<td>lod0Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>LOD</td>
<td>Property Type</td>
<td>minOccurs</td>
<td>maxOccurs</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>0</td>
<td>gml:GeometryPropertyType</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>gml:GeometryPropertyType</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>gml:GeometryPropertyType</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>gml:GeometryPropertyType</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>gml:GeometryPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>ImplicitRepresentationPropertyType</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>ImplicitRepresentationPropertyType</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>ImplicitRepresentationPropertyType</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ImplicitRepresentationPropertyType</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>ImplicitRepresentationPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**CityObjectGroupType**

- function: xs:string (0 – 4)
- class: xs:string (0 – 4)
- usage: xs:string (0 – 4)
- groupMember: CityObjectGroupMemberType (0 – 4)
- geometry: gml:GeometryPropertyType (0 – 4)

**AbstractBuildingType**

- class: BuildingClassType (1 – 4)
- function: BuildingFunctionType (1 – 4)
- usage: BuildingUsageType (1 – 4)
- yearOfConstruction: xs:gYear (1 – 4)
- roofType: RoofTypeType (1 – 4)
- measuredHeight: gml:LengthType (1 – 4)
- storeysAboveGround: xs:nonNegativeInteger (1 – 4)
- storeysBelowGround: xs:nonNegativeInteger (1 – 4)
- storeyHeightsAboveGround: gml:MeasureOrNullListType (1 – 4)
- storeyHeightsBelowGround: gml:MeasureOrNullListType (1 – 4)
- lod1Solid: gml:SolidPropertyType (1)
- lod1MultiSurface: gml:MultiSurfacePropertyType (1)
- lod1MultiCurve: gml:MultiCurvePropertyType (1)
- lod1TerrainIntersection: gml:MultiCurvePropertyType (1)
- lod2Solid: gml:SolidPropertyType (2)
- lod2MultiSurface: gml:MultiSurfacePropertyType (2)
- lod2MultiCurve: gml:MultiCurvePropertyType (2)
- lod2TerrainIntersection: gml:MultiCurvePropertyType (2)
- outerBuildingInstallation: BuildingInstallationPropertyType (1 – 4)
- boundedBy: BoundarySurfacePropertyType (2 – 4)
- lod3Solid: gml:SolidPropertyType (2)
- lod3MultiSurface: gml:MultiSurfacePropertyType (2)
- lod3MultiCurve: gml:MultiCurvePropertyType (2)
- lod3TerrainIntersection: gml:MultiCurvePropertyType (2)
- lod4Solid: gml:SolidPropertyType (4)
- lod4MultiSurface: gml:MultiSurfacePropertyType (4)
- lod4MultiCurve: gml:MultiCurvePropertyType (4)
- lod4TerrainIntersection: gml:MultiCurvePropertyType (4)
- interiorRoom: InteriorRoomPropertyType (4)
- consistsOfBuildingPart: BuildingPartPropertyType (1 – 4)
- address: AddressPropertyType (1 – 4)

**BuildingPartType**

- class: BuildingPartClassType (1 – 4)
- function: BuildingPartFunctionType (1 – 4)
- usage: BuildingPartUsageType (1 – 4)
- lod2Geometry: gml:GeometryPropertyType (2)
- lod3Geometry: gml:GeometryPropertyType (3)
- lod4Geometry: gml:GeometryPropertyType (4)
- opening: OpeningPropertyType (3 – 4)

**BoundarySurfaceType**

- lod2MultiSurface: gml:MultiSurfacePropertyType (2)
- lod3MultiSurface: gml:MultiSurfacePropertyType (3)
- lod4MultiSurface: gml:MultiSurfacePropertyType (4)

**RoofSurfaceType**

- opening: OpeningPropertyType (3 – 4)

**WallSurfaceType**

- opening: OpeningPropertyType (3 – 4)
<p>| GroundSurfaceType | 2 – 4 |
| ClosureSurfaceType | 2 – 4 |
| FloorSurfaceType | 4 |
| InteriorWallSurfaceType | 4 |
| CeilingSurfaceType | 4 |
| OpeningType | 3 – 4 |
| WindowType | 3 – 4 |
| DoorType | 3 – 4 |
| RoomType | 4 |
| BuildingFurnitureType | 4 |
| TransportationObjectType | 0 – 4 |
| TrafficAreaType | 1 – 4 |
| AuxiliaryTrafficAreaType | 1 – 4 |
| TrackType | 1 – 4 |
| RoadType | 1 – 4 |
| RailwayType | 1 – 4 |
| SquareType | 1 – 4 |
| VegetationObjectType | 1 – 4 |
| PlantCoverType | 1 – 4 |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lod1MultiSolid</td>
<td>gml:MultiSolidPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2MultiSolid</td>
<td>gml:MultiSolidPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3MultiSolid</td>
<td>gml:MultiSolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>SolitaryVegetationObject</td>
<td></td>
<td>1 – 4</td>
</tr>
<tr>
<td>class</td>
<td>PlantClassType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>function</td>
<td>PlantFunctionType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>species</td>
<td>SpeciesType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>height</td>
<td>gml:LengthType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>trunkDiameter</td>
<td>gml:LengthType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>crownDiameter</td>
<td>gml:LengthType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>WaterObjectType</td>
<td></td>
<td>0 – 4</td>
</tr>
<tr>
<td>ImplicitGeometryType</td>
<td></td>
<td>0 – 4</td>
</tr>
<tr>
<td>mimeType</td>
<td>MimeTypeType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>transformationMatrix</td>
<td>TransformationMatrixType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>libraryObject</td>
<td>xs:anyURI</td>
<td>0 – 4</td>
</tr>
<tr>
<td>relativeGMLGeometry</td>
<td>gml:GeometryPropertyType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>referencePoint</td>
<td>gml:PointPropertyType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>WaterBodyType</td>
<td></td>
<td>0 – 4</td>
</tr>
<tr>
<td>class</td>
<td>WaterBodyClassType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>function</td>
<td>WaterBodyFunctionType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>WaterBodyUsageType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>lod0MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>0</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod1Solid</td>
<td>gml:SolidPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2Solid</td>
<td>gml:SolidPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3Solid</td>
<td>gml:SolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>BoundedByWaterSurfacePropertyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>WaterBoundarySurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td>lod2Surface</td>
<td>gml:SurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3Surface</td>
<td>gml:SurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Surface</td>
<td>gml:SurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>WaterSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td>waterLevel</td>
<td>WaterLevelType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>WaterGroundSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td>WaterClosureSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td>LandUseType</td>
<td></td>
<td>0 – 4</td>
</tr>
<tr>
<td>class</td>
<td>LandUseClassType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>function</td>
<td>LandUseFunctionType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>LandUseUsageType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>0</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>lod ImplicitRepresentation</td>
<td>ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>ReliefFeatureType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod</td>
<td>integerBetween0and4</td>
</tr>
<tr>
<td></td>
<td>reliefComponent</td>
<td>ReliefComponentPropertyType</td>
</tr>
<tr>
<td>ReliefComponentType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod</td>
<td>integerBetween0and4</td>
</tr>
<tr>
<td></td>
<td>extent</td>
<td>gml:PolygonPropertyType</td>
</tr>
<tr>
<td>TINReliefType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tin</td>
<td>tinPropertyType</td>
</tr>
<tr>
<td>RasterReliefType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>grid</td>
<td>gridPropertyType</td>
</tr>
<tr>
<td>MassPointReliefType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reliefPoints</td>
<td>gml:MultiPointPropertyType</td>
</tr>
<tr>
<td>MassPointRelief</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BreakLineReliefType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ridgeOrValleyLines</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td></td>
<td>breaklines</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
</tbody>
</table>
11.4 Annex D: Examples

11.4.1 Example of a CityGML dataset for a building in LOD 1 and 2

Fig. 44: Visualisation of the following CityGML dataset containing buildings in LOD1 and 2 (source: IKG Uni Bonn).

<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.citygml.org/citygml/1/0/0" xmlns:gml="http://www.opengis.net/gml"
xsi:schemaLocation="http://www.citygml.org/citygml/1/0/0 ..CityGML.xsd">
  <gml:description>Simple example for an XML dataset according to CityGML, the GML application schema of the SIG 3D. This dataset contains four parts with different complexities, which have been truncated here (the full version can be obtained from www.citygml.org):
1.) Simple building in LOD2 with one textured and one colored surface
2.) Simple building in LOD1 as blocks model without balcony, and the same building with gabled roof and balcony in LOD2.
3.) House with gabled roof and garage, represented by two BuildingParts. The common wall surface of the building and the garage is defined only once and is in the boundary of one solid, and re-used by the second solid.
4.) Building group consisting of two buildings that have been defined previously.
The coordinate reference system is given in ETRS89 / Gauss-Krüger 3 degree (2nd zone) + ellipsoidal elevation. This system is referred to by srsName="urn:adv:crs: ETRS89_3GK2-h". Please note that the coordinates actually used in this dataset have been trimmed for clarity reasons and thus do not match this CRS.
</gml:description>
  <gml:name>3D city model of Samplecity</gml:name>
  <gml:boundedBy>
    <gml:Envelope srsName="urn:adv:crs: ETRS89_3GK2-h">
      <gml:pos srsDimension="3">0.0 0.0 0.0</gml:pos>
      <gml:pos srsDimension="3">33.0 34.0 2.5</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <cityObjectMember>
    <!--Simple building with gabled roof with two storeys and an address. It is a LOD 2 model, because it contains a roof shape.-->
    <Building gml:id="Build0815">
      <externalReference>
        <informationSystem>http://www.adv-online.de</informationSystem>
        <!-- Reference to the german cadastral database -->
        <externalObject>
          <uri>urn:adv:oid:DEHE1234000007001</uri>
          <!-- 1D of the object, being unique country-wide -->
        </externalObject>
      </externalReference>
      <function>1000</function>
      <yearOfConstruction>1985</yearOfConstruction>
      <roofType>1030</roofType>
      <measuredHeight uom="#m">8.0</measuredHeight>
      <storeysAboveGround>2</storeysAboveGround>
      <storeyHeightsAboveGround uom="#m">2.5 2.5</storeyHeightsAboveGround>
      <lod2Solid>
        <!--simple building with gabled roof-->
        <gml:Solid srsName="urn:adv:crs: ETRS89_3GK2-h">
          <gml:exterior>
            <gml:CompositeSurface>
              <gml:surfaceMember>
                109
              </gml:surfaceMember>
            </gml:CompositeSurface>
          </gml:exterior>
        </gml:Solid>
      </lod2Solid>
    </Building>
  </cityObjectMember>
</CityModel>
<TexturedSurface orientation="+">
  <!-- front surface -->
  <gml:baseSurface>
    <gml:Polygon>
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList srsDimension="3">
            1.0 1.0 0.0
            3.0 1.0 1.5
            2.0 1.0 2.5
            1.0 1.0 1.5
            1.0 1.0 0.0
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
    <appearance>
      <SimpleTexture>
        <textureMap>FrontTexture096454.jpg</textureMap>
        <textureCoordinates>0.05 0.07 0.95 0.07 0.95 0.5 0.5 1 0.05 0.5 0.05 0.07</textureCoordinates>
        <textureType>specific</textureType>
      </SimpleTexture>
    </appearance>
  </gml:baseSurface>
</TexturedSurface>

<TexturedSurface orientation="+">
  <!-- back surface -->
  <gml:baseSurface>
    <gml:Polygon>
      <gml:exterior>
        <gml:LinearRing>
          <gml:pos srsDimension="3">1.0 4.0 0.0</gml:pos>
          <gml:pos srsDimension="3">1.0 4.0 1.5</gml:pos>
          <gml:pos srsDimension="3">2.0 4.0 2.5</gml:pos>
          <gml:pos srsDimension="3">3.0 4.0 1.5</gml:pos>
          <gml:pos srsDimension="3">3.0 4.0 0.0</gml:pos>
          <gml:pos srsDimension="3">1.0 4.0 0.0</gml:pos>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
    <appearance>
      <Material>
        <ambientIntensity>0.4</ambientIntensity>
        <diffuseColor>0 0 1</diffuseColor> <!-- defines blue color -->
      </Material>
    </appearance>
  </gml:baseSurface>
</TexturedSurface>

<Building gml:id="Build0816">
  <gml:name>Villa Kunterbunt</gml:name>
  <function>1000</function>
  <yearOfConstruction>1952</yearOfConstruction>
  <roofType>1030</roofType>
  <lod1Solid>
    <gml:CompositeSurface>
      <!-- simple blocks model without balcony -->
      <gml:exterior>
        <gml:Polygons>
          <gml:exterior>
            <gml:LinearRing>
              <gml:posList srsDimension="3">
                1.0 4.0 0.0
                3.0 4.0 1.5
                2.0 4.0 2.5
                1.0 4.0 1.5
                1.0 4.0 0.0
              </gml:posList>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygons>
        <appearance>
          <SimpleTexture>
            <textureMap>FrontTexture096454.jpg</textureMap>
            <textureCoordinates>0.05 0.07 0.95 0.07 0.95 0.5 0.5 1 0.05 0.5 0.05 0.07</textureCoordinates>
            <textureType>specific</textureType>
          </SimpleTexture>
        </appearance>
      </gml:exterior>
    </gml:CompositeSurface>
  </lod1Solid>
</Building>

<cityObjectMember>
  <!-- simple building represented in LOD1 (as blocks model without balcony) and in LOD2
       with roof shape and balcony. One of the roof surfaces is represented explicitly as a thematic surface object (RoofSurface).
       The function is residential building (1000) and the roof type is 'gabled roof' (1030).
       Both values are defined in external code lists. -->
</cityObjectMember>
<gml:Polygon>
  <gml:exterior>
    <gml:LinearRing>
      <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
      <gml:pos srsDimension="3">33.0 31.0 0.0</gml:pos>
      <gml:pos srsDimension="3">33.0 31.0 1.5</gml:pos>
      <gml:pos srsDimension="3">31.0 31.0 1.5</gml:pos>
      <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
    </gml:LinearRing>
  </gml:exterior>
</gml:Polygon>

..................................
</gml:surfaceMember>
</gml:CompositeSurface>
</gml:exterior>
</lod1Solid>
</lod2Solid>
</gml:Solid srsName="urn:adv:crs: ETRS89_3GK2-h">
<!-- simple building with gabled roof -->
<gml:exterior>
  <gml:CompositeSurface>
    <gml:surfaceMember>
      <!--front surface -->
      <gml:Polygon>
        <gml:exterior>
          <gml:LinearRing>
            <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
            <gml:pos srsDimension="3">33.0 31.0 0.0</gml:pos>
            <gml:pos srsDimension="3">33.0 31.0 1.5</gml:pos>
            <gml:pos srsDimension="3">32.0 31.0 2.5</gml:pos>
            <gml:pos srsDimension="3">31.0 31.0 1.5</gml:pos>
            <gml:pos srsDimension="3">31.0 31.0 0.0</gml:pos>
          </gml:LinearRing>
        </gml:exterior>
      </gml:Polygon>
      .........................................
    </gml:surfaceMember>
    <gml:surfaceMember>
      <!--1st roof surface. This polygon will be referenced below -->
      <gml:Polygon gml:id="roofsurface4711">
        <gml:exterior>
          <gml:LinearRing>
            <gml:posList srsDimension="3">
              32.0 31.0 2.5
              33.0 31.0 1.5
              33.0 34.0 1.5
              32.0 34.0 2.5
              32.0 31.0 2.5
            </gml:posList>
          </gml:LinearRing>
        </gml:exterior>
        ..........................................
      </gml:Polygon>
    </gml:surfaceMember>
  </gml:CompositeSurface>
</gml:exterior>
</gml:Solid>
</outerBuildingInstallation>
<BuildingInstallation>
  <gml:name>The nice balcony to the south</gml:name>
  <gml:function>1000</gml:function>
  <!--function 1000 of a BuildingInstallation means 'balcony'-->
  <lod2Geometry>
    <!--The balcony is situated at the 1st front surface-->
    <!--The geometry of the balcony is defined by an aggregation of 3D surfaces.-->
    <gml:CompositeSurface srsName="urn:adv:crs: ETRS89_3GK2-h">
      <gml:surfaceMember>
        <!-- ground surface of the balcony -->
        <gml:Polygon>
          <gml:exterior>
            <gml:LinearRing>
              <gml:pos srsDimension="3">31.5 30.5 0.8</gml:pos>
              <gml:pos srsDimension="3">31.5 31.0 0.8</gml:pos>
              <gml:pos srsDimension="3">32.5 31.0 0.8</gml:pos>
              <gml:pos srsDimension="3">32.5 30.5 0.8</gml:pos>
              <gml:pos srsDimension="3">31.5 30.5 0.8</gml:pos>
            </gml:LinearRing>
          </gml:exterior>
        </gml:Polygon>
      </gml:surfaceMember>
      ..........................................
    </gml:CompositeSurface>
</gml:exterior>
</gml:Solid>
</BuildingInstallation>
House with gabled roof and a garage, represented by two BuildingParts. The common wall surface of the building and the garage is shared by both solids realizing a topological connection between both parts.

Building gml:id="Build0817">
  consistsOfBuildingPart>
    BuildingPart gml:id="Build0817a">
      function>1000</function>
      yearOfConstruction>1964</yearOfConstruction>
      roofType>1030</roofType>
      storeysAboveGround>2</storeysAboveGround>
      lod2Solid">
        gml:Solid srsName="urn:advers:ETRS89_3GK2-h">
          !<Building with gabled roof->
            gml:exterior>
              gml:CompositeSurface>
                gml:surfaceMember>
                  !<front surface->
                    gml:Polygon>
                      gml:LinearRing>
                        gml:posList srsDimension="3">
                          8.0 2.0 0.0
                          8.0 4.0 0.0
                          8.0 4.0 1.5
                          8.0 3.0 2.5
                          8.0 2.0 1.5
                          8.0 2.0 0.0
                      </gml:posList>
                    </gml:LinearRing>
                  </gml:exterior>
                </gml:Polygon>
                gml:Polygon>
                  gml:LinearRing>
                    gml:posList srsDimension="3">
                      8.0 4.0 1.0
                      8.0 4.0 0.0
                    </gml:posList>
                  </gml:LinearRing>
                </gml:Polygon>
                gml:Polygon>
                  gml:LinearRing>
                    gml:posList srsDimension="3">
                      6.5 4.0 1.0
                      8.0 4.0 1.0
                    </gml:posList>
                  </gml:LinearRing>
                </gml:Polygon>
              </gml:surfaceMember>
            </gml:Polygon>
          </gml:exterior>
        </gml:Polygon>
      </gml:surfaceMember>
    </BuildingPart>
  </consistsOfBuildingPart>
</Building>
Listing 1: Excerpt from the CityGML dataset for buildings in LOD1 and 2 visualised in Fig. 44.
11.4.2 Example of a CityGML dataset for a building in LOD 3

Fig. 45: Visualisation of buildings in LOD 3, automatically generated from IFC building objects. Please note the coherent semantic and geometric decomposition (source: Research Center Karlsruhe).

    <gml:description>This file contains four buildings which are automatically converted from IFC models. This listing only shows an excerpt. The full dataset can be downloaded from http://www.citygml.org (example dataset for “four buildings in LOD3”).</gml:description>  
    <gml:name>IFC_Building_Variant</gml:name>  
    <gml:boundedBy>  
        <gml:Envelope srsName="EPSG:31467">  
            <gml:pos srsDimension="3">3449999.751795 5429999.751795 0.0</gml:pos>  
            <gml:pos srsDimension="3">3450021.2 5430023.2 20.0</gml:pos>  
        </gml:Envelope>  
    </gml:boundedBy>  
    <cityObjectMember>  
        <Building gml:id="GEB_TH_IFC_Building_Variant_GEB_75">  
            <gml:description>Building in LOD 3</gml:description>  
            <gml:name>Building-ADT-2006</gml:name>  
            <externalReference>  
                <informationSystem>IFC</informationSystem>  
                <externalObject>  
                    <uri>urn:ifc:oid: 0deJpNQ05BvwV03c405oVp</uri>  
                </externalObject>  
            </externalReference>  
            <boundedBy>  
                <RoofSurface gml:id="GEB_TH_IFC_Building_Variant_DACH_136">  
                    <externalReference>  
                        <informationSystem>IFC</informationSystem>  
                        <externalObject>  
                            <uri>urn:ifc:oid: 3CPSkwS7f9QRfhfr5gf7dq</uri>  
                        </externalObject>  
                    </externalReference>  
                    <lod3MultiSurface>  
                        <gml:MultiSurface>  
                            <gml:surfaceMember>  
                                <gml:Polygon>  
                                    <gml:exterior>  
                                        <gml:LinearRing>  
                                        </gml:LinearRing>  
                                    </gml:exterior>  
                                </gml:Polygon>  
                            </gml:surfaceMember>  
                        </gml:MultiSurface>  
                    </lod3MultiSurface>  
                </RoofSurface>  
            </boundedBy>  
        </Building>  
    </cityObjectMember>  
</CityModel>
Listing 2: Excerpt from the CityGML dataset for the buildings in LOD3 visualised in Fig. 45.
12 Bibliography


