OGC City Geography Markup Language (CityGML) Encoding Standard

Copyright © 2012 Open Geospatial Consortium.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. 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.
License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.
Contents

i. Abstract ........................................................................................................... ix
ii. Preface and Acknowledgements ................................................................. ix
iii. Submitting organizations .......................................................................... ix
iv. Submission contact points .......................................................................... x
v. Participants in development ........................................................................ x
vi. Changes to the OGC® Abstract Specification ............................................ xi
vii. Acknowledgments ....................................................................................... xi

Foreword ............................................................................................................ xii

0 Introduction .................................................................................................... xiv
  0.1 Motivation ..................................................................................................... xiv
  0.2 Historical background .............................................................................. xiv
  0.3 Additions in CityGML 2.0 ......................................................................... xv

1 Scope .............................................................................................................. 1

2 Conformance .................................................................................................. 2

3 Normative references ................................................................................... 2

4 Conventions .................................................................................................... 3
  4.1 Abbreviated terms ...................................................................................... 3
  4.2 UML Notation ............................................................................................. 4
  4.3 XML namespaces and namespace prefixes ............................................. 6
  4.4 XML-Schema ............................................................................................. 7

5 Overview of CityGML .................................................................................... 9

6 General characteristics of CityGML ........................................................... 11
  6.1 Modularisation ........................................................................................... 11
  6.2 Multi-scale modelling (5 levels of detail, LOD) ....................................... 11
  6.3 Coherent semantical-geometrical modelling ........................................... 12
  6.4 Closure surfaces ....................................................................................... 12
  6.5 Terrain Intersection Curve (TIC) ............................................................. 13
  6.6 Code lists for enumerative attributes ...................................................... 14
  6.7 External references .................................................................................... 14
  6.8 City object groups ..................................................................................... 14
  6.9 Appearances ............................................................................................. 15
  6.10 Prototypic objects / scene graph concepts ............................................. 15
  6.11 Generic city objects and attributes ......................................................... 15
  6.12 Application Domain Extensions (ADE) ................................................ 16
7 Modularisation ........................................................................................................................................... 17
  7.1 CityGML core and extension modules ........................................................................................... 18
  7.2 CityGML profiles ............................................................................................................................ 23
8 Spatial model ........................................................................................................................................... 23
  8.1 Geometric-topological model ......................................................................................................... 23
  8.2 Spatial reference system ................................................................................................................ 27
  8.3 Implicit geometries, prototypic objects, scene graph concepts ...................................................... 28
    8.3.1 Code lists ................................................................................................................................. 30
    8.3.2 Example CityGML datasets .................................................................................................. 30
    8.3.3 Conformance requirements .................................................................................................... 31
9 Appearance model .................................................................................................................................. 31
  9.1 Relation between appearances, features and geometry ................................................................. 33
  9.2 Appearance and SurfaceData ........................................................................................................ 36
  9.3 Material ........................................................................................................................................... 37
  9.4 Texture and texture mapping ........................................................................................................ 38
  9.5 Related concepts ........................................................................................................................... 44
  9.6 Code lists ......................................................................................................................................... 44
  9.7 Conformance requirements ........................................................................................................... 44
  9.8 Material model of previous CityGML versions [deprecated] ......................................................... 47
    9.8.1 Textured surfaces ................................................................................................................... 47
    9.8.2 Conformance requirements ................................................................................................ 48
10 Thematic model ....................................................................................................................................... 49
  10.1 CityGML Core ............................................................................................................................... 50
    10.1.1 Base elements ........................................................................................................................ 52
    10.1.2 Generalisation relation, RelativeToTerrainType and RelativeToWaterType ...................... 53
    10.1.3 External references ............................................................................................................... 54
    10.1.4 Address information ............................................................................................................ 54
    10.1.5 Code lists ................................................................................................................................ 56
    10.1.6 Conformance requirements .................................................................................................. 56
  10.2 Digital Terrain Model (DTM) ........................................................................................................ 57
    10.2.1 Relief feature and relief component ....................................................................................... 58
    10.2.2 TIN relief ............................................................................................................................... 59
    10.2.3 Raster relief .......................................................................................................................... 59
    10.2.4 Mass point relief ................................................................................................................... 60
    10.2.5 Breakline relief .................................................................................................................... 60
    10.2.6 Conformance requirements .................................................................................................. 61
  10.3 Building model ............................................................................................................................... 62
    10.3.1 Building and building part..................................................................................................... 64
    10.3.2 Outer building installations ................................................................................................... 68
    10.3.3 Boundary surfaces ................................................................................................................. 69
Annex C (informative)  Code lists proposed by the SIG 3D ......................... 223
C.1  Building module .................................................. 226
C.2  Tunnel module ................................................... 233
C.3  Bridge module .................................................. 234
C.4  CityFurniture module ......................................... 235
C.5  LandUse module .................................................. 236
C.6  Mime types ....................................................... 237
C.7  Vegetation module .................................................. 238
C.8  Transportation module ........................................... 240
C.9  WaterBody module .................................................. 243
C.10 CityObjectGroup module .......................................... 245

Annex D (informative)  Overview of employed GML3 geometry classes .......... 247
Annex E (informative)  Overview of the assignment of features to LODs ........... 249
Annex F (informative)  Changelog for CityGML 2.0 ............................... 261
Annex G (informative)  Example CityGML datasets .................................... 269
G.1  Example of a CityGML dataset for a building in LOD0 ..................... 269
G.2  Example of a CityGML dataset for a building in LOD1 ..................... 272
G.3  Example of a CityGML dataset for a building in LOD2 ..................... 275
G.4 Example of a CityGML dataset for a building in LOD2 with an adjacent building part illustrating CityGML’s topology representation ........................................ 278
G.5 Example of a CityGML dataset for a building in LOD3 ........................................ 282
G.6 Example of a CityGML dataset for a building in LOD4 ........................................ 286
G.7 Example of a CityGML dataset illustrating the appearance model .................. 291
G.8 Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes ................................................................. 298
G.9 Example of a CityGML dataset illustrating the use of local coordinate reference systems ................................................................................................. 301
Annex H (informative) Example ADE for Noise Immission Simulation .................... 305
  H.1 CityGML Noise ADE .......................................................................................... 308
  H.2 Example dataset .............................................................................................. 312
Annex I (informative) Example ADE for Ubiquitous Network Robots Services ..... 315
  I.1 Overview of Ubiquitous Network Robots ...................................................... 315
  I.2 Overview of the Spatial Master Database .................................................... 317
  I.3 Overview of the CityGML ADE ...................................................................... 318
  I.4 Example Dataset .......................................................................................... 321
Annex K (informative) Revision history ................................................................ 324
Bibliography .......................................................................................................... 325
i. Abstract

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211. The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

ii. Preface and Acknowledgements

This is the official CityGML logo. For current news on CityGML and information about ongoing projects and fields of research in the area of CityGML see http://www.citygml.org and http://www.citygmlwiki.org

OGC work on CityGML is discussed and coordinated by the OGC 3D Information Management (3DIM) Working Group. CityGML was initially implemented and evaluated as part of the OGC Web Services Testbed, Phase 4 (OWS-4) in the CAD/GIS/BIM thread.

Version 2.0 of this standards document was prepared by the OGC CityGML Standards Working Group (SWG). Future discussion and development will be led by the 3DIM Working Group.

For further information see http://www.opengeospatial.org/projects/groups/3dimwg

CityGML also continues to be developed by the members of the Special Interest Group 3D (SIG 3D) of the GDI-DE Geodateninfrastruktur Deutschland (Spatial Data Infrastructure Germany) in joint cooperation with the 3DIM Working Group and the CityGML SWG within OGC.

For further information see http://www.sig3d.org/

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 an EuroSDR Commission III project.

For further information see http://www.eurosdr.net

iii. Submitting organizations

This International Standard was submitted to the Open Geospatial Consortium Inc. by the members of the CityGML 1.0 Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

a) Autodesk, Inc. (primary submitter)
b) Bentley Systems, Inc. (primary submitter)
c) Technical University Berlin (submitter of technology)
d) Ordnance Survey, UK
CityGML was originally developed by the Special Interest Group 3D (SIG 3D), 2002 – 2012 - www.citygml.org.

iv. Submission contact points

All questions regarding this document should be directed to the editors or the contributors (including participants in development, cf. clause v):

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prof. Dr. Thomas H. Kolbe</td>
<td>Institute for Geodesy and Geoinformation Science, Technical University Berlin</td>
<td><a href="mailto:thomas.kolbe@tu-berlin.de">thomas.kolbe@tu-berlin.de</a></td>
</tr>
<tr>
<td>Claus Nagel</td>
<td></td>
<td><a href="mailto:claus.nagel@tu-berlin.de">claus.nagel@tu-berlin.de</a></td>
</tr>
<tr>
<td>Alexandra Lorenz</td>
<td></td>
<td><a href="mailto:alexandra.lorenz@tu-berlin.de">alexandra.lorenz@tu-berlin.de</a></td>
</tr>
<tr>
<td>Dr. Gerhard Gröger</td>
<td>Institute for Geodesy and Geoinformation, University of Bonn</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>Haik Lorenz</td>
<td>Autodesk, Inc.</td>
<td><a href="mailto:haik.lorenz@autodesk.com">haik.lorenz@autodesk.com</a></td>
</tr>
<tr>
<td>Alain Lapierre</td>
<td>Bentley Systems, Inc.</td>
<td><a href="mailto:alain.lapierre@bentley.com">alain.lapierre@bentley.com</a></td>
</tr>
<tr>
<td>Stefan Apfel</td>
<td></td>
<td><a href="mailto:stefan.apfel@bentley.com">stefan.apfel@bentley.com</a></td>
</tr>
<tr>
<td>Paul Scarponcini</td>
<td></td>
<td><a href="mailto:paul.scarponcini@bentley.com">paul.scarponcini@bentley.com</a></td>
</tr>
<tr>
<td>Carsten Rönisdorf</td>
<td>Ordnance Survey, Great Britain</td>
<td><a href="mailto:carsten.roensdorf@ordnancesurvey.co.uk">carsten.roensdorf@ordnancesurvey.co.uk</a></td>
</tr>
<tr>
<td>Prof. Dr. Jürgen Dollner</td>
<td>Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam</td>
<td><a href="mailto:juergen.dollner@hpi.uni-potsdam.de">juergen.dollner@hpi.uni-potsdam.de</a></td>
</tr>
<tr>
<td>Dr. Joachim Benner</td>
<td>Institute for Applied Computer Science, Karlsruhe Institute of Technology</td>
<td><a href="mailto:joachim.benner@kit.edu">joachim.benner@kit.edu</a></td>
</tr>
<tr>
<td>Karl-Heinz Hafele</td>
<td></td>
<td><a href="mailto:karl-heinz.hafele@kit.edu">karl-heinz.hafele@kit.edu</a></td>
</tr>
</tbody>
</table>

v. Participants in development

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ulrich Gruber, Sandra Schlüter</td>
<td>District Administration Recklinghausen, Cadastre Department, Germany</td>
</tr>
<tr>
<td>Frank Bildstein</td>
<td>Rheinmetall Defence Electronics, Germany</td>
</tr>
<tr>
<td>Rüdiger Drees</td>
<td>T-Systems Enterprise Services GmbH, Bonn, Germany</td>
</tr>
<tr>
<td>Andreas Kohlhaas</td>
<td>GISTEC GmbH (formerly), Germany</td>
</tr>
<tr>
<td>Frank Thiemann</td>
<td>Institute for Cartography and Geoinformatics, University of Hannover</td>
</tr>
<tr>
<td>Martin Degen</td>
<td>Cadastre Department, City of Dortmund</td>
</tr>
<tr>
<td>Heinrich Geerling</td>
<td>Architekturbüro Geerling, Germany</td>
</tr>
<tr>
<td>Dr. Frank Knospe</td>
<td>Cadastre and Mapping Department, City of Essen,</td>
</tr>
<tr>
<td>Hardo Müller</td>
<td>Snowflake Software Ltd., Great Britain</td>
</tr>
<tr>
<td>Martin Recliner</td>
<td>rlechner logistic, Germany</td>
</tr>
<tr>
<td>Jörg Haist</td>
<td>Fraunhofer Institute for Computer Graphics (IGD), Darmstadt, Germany</td>
</tr>
<tr>
<td>Daniel Holweg</td>
<td></td>
</tr>
<tr>
<td>Prof. Dr. Peter A. Henning</td>
<td>Faculty for Computer Science, University of Applied Sciences, Karlsruhe, Germany</td>
</tr>
<tr>
<td>Rolf Wegener</td>
<td>State Cadastre and Mapping Agency of North-Rhine Westphalia, Germany</td>
</tr>
<tr>
<td>Stephan Heinmann</td>
<td></td>
</tr>
<tr>
<td>Prof. Dr. Marc-O. Löwner</td>
<td>Institute for Geodesy and Photogrammetry, Technical University of Braunschweig</td>
</tr>
<tr>
<td>Dr. Egbert Casper</td>
<td>Zerna Ingenieure, Germany</td>
</tr>
<tr>
<td>Christian Dahmen</td>
<td>con terra GmbH, Germany</td>
</tr>
<tr>
<td>Nobuhiro Ishimaru</td>
<td>Hitachi, Ltd., Japan</td>
</tr>
</tbody>
</table>
vi. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

vii. Acknowledgments

The SIG 3D wishes to thank the members of the CityGML Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC as well as all contributors of change requests and comments. In particular: Tim Case, Scott Simmons, Paul Cote, Clemens Portele, Jeffrey Bell, Chris Body, Greg Buehler, François Golay, John Herring, Jury Konga, Kai-Uwe Krause, Gavin Park, Richard Pearsall, George Percivall, Mauro Salvemini, Alessandro Triglia, David Wesloh, Tim Wilson, Greg Yetman, Jim Farley, Cliff Behrens, Lukas Herman, Danny Kita, and Simon Cox.

Further credits for careful reviewing and commenting of this document go to: Ludvig Emgard, Bettina Petzold, Dave Capstick, Mark Pendlington, Alain Lapierre, and Frank Steggink.
Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. However, to date, no such rights have been claimed or identified.

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

Significant changes between CityGML version 2.0.0 and CityGML version 1.0.0 (OGC document no. 08-007r1):

- New thematic modules for the representation of tunnels and bridges;
- Additional boundary surfaces for the semantic classification of the outer shell of buildings and building parts (OuterCeilingSurface and OuterFloorSurface);
- LOD0 representation (footprint and roof edge representations) for buildings and building parts;
- Additional attributes denoting a city object’s location with respect to the surrounding terrain and water surface (relativeToTerrain and relativeToWater);
- Additional generic attributes for measured values and attribute sets; and
- Redesign of the CityGML code list mechanism (enumerative attributes are now of type gml:CodeType which facilitates to provide additional code lists enumerating their possible attribute values).

Migration of existing CityGML 1.0 instances to valid 2.0 instances only requires changing the CityGML namespace and schema location values in the document to the actual 2.0 values.
0 Introduction

0.1 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 and 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 visualisation 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 had 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, tourist 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 extendible 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 generalisation 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 either simple, single scale models without topology and few semantics or 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.

0.2 Historical background

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D). Since 2010, this group is part of the initiative Spatial Data Infrastructure Germany (GDI-DE). Before 2010, the SIG 3D was affiliated to the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW). 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 city models and geovisualisation. 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 and OGC Doc. No. 09-104r1).

A first successful implementation and evaluation of a subset of CityGML has been performed in the project “Pilot 3D” of the GDI NRW in 2005. Participants came from all over Germany and demonstrated city planning scenarios and tourist applications. By the beginning of 2006, a CityGML project within EuroSDR (European Spatial Data Research) started focusing on the European harmonisation of 3D city modelling. From June to December 2006, CityGML was employed and evaluated in the CAD/GIS/BIM thread of the OpenGIS Web Services Testbed #4 (OWS-4). Since 2008, CityGML (version 1.0.0) is an adopted OGC standard.
From that point in time, CityGML has disseminated worldwide. Many cities in Germany and in other countries in Europe provide their 3D city model in CityGML (Berlin, Cologne, Dresden and Munich, to mention only a few). In France, the project Bâti3D (IGN France) defines a profile of CityGML LOD2 and provides data from Paris and the city centres of Aix-en-Provence, Lille, Nantes and Marseille. CityGML also plays an important role in the pilot 3D project to obtain a 3D geoinformation standard and a 3D infrastructure for The Netherlands. Many cities in Europe like Monaco, Geneva, Zurich, Leewarden use CityGML LOD 2 or 3 to represent and exchange data, as well as cities in Denmark (LOD 2 and 3, partly LOD4). CityGML has strongly influenced the building model (version 2.0) of the INSPIRE initiative of the EU commission, which aims at the creation of an European spatial data infrastructure providing public sector data in an interoperable way. In Asia, the 3D city models of Istanbul (LOD 1 and 2), Doha, Katar (LOD3), and Yokohama (LOD2) are represented and exchanged in CityGML. Moreover, CityGML plays a crucial role for the 3D Spatial data infrastructure in Malaysia.

Today many commercial and academic tools support CityGML by providing import interfaces, export interfaces or both. An example is the 3D City Database which is a free and open source 3D geo database to store, represent, and manage virtual 3D city models on top of Oracle 10g R2 and 11g R1/R2 provided by the Technische Universität Berlin. It fully supports CityGML and is shipped with a tool for the import and export of CityGML models. Furthermore, an open source Java class library and API for the processing of CityGML models (citygml4j) is provided by the Technische Universität Berlin. The conversion tool FME (Feature Manipulation Engine) from Safe Software Inc., which is part of the interoperability extension of ESRI’s ArcGIS, has read and write interfaces for CityGML. The same applies to CAD tools as BentleyMap from Bentley Systems as well as to GIS tools like SupportGIS from CPA Geo-Information. Many 3D viewers (which all are freely available) provide read interfaces for CityGML: the Aristoteles Viewer from the University of Bonn, LandXplorer CityGML Viewer from Autodesk Inc. (the studio version for authoring and management is not free) and the FZKviewer for IFC and CityGML from KIT Karlsruhe and BS Contact from Bitmanagement Software GmbH which offers a CityGML plugin for the geospatial extension BS Contact Geo. This enumeration of software tools is not exhaustive and steadily growing. Please refer to the official website of CityGML at http://www.citygml.org as well as the CityGML Wiki at http://www.citygmlwiki.org for more information.

### 0.3 Additions in CityGML 2.0

CityGML 2.0 is a major revision of the previous version 1.0 of this International Standard (OGC Doc. No. 08-007r1), and introduces substantial additions and new features to the thematic model of CityGML. The revision was originally planned to be a minor update to version 1.1. The main endeavor of the revision process was to ensure backwards compatibility both on the level of the conceptual model and on the level of CityGML instance documents. However, some changes could not be implemented consistent with directives for minor revisions and backwards compatibility as enforced by OGC policy (cf. OGC Doc. No. 135r11). The major version number change to 2.0 is therefore a consequence of conforming to the OGC versioning policy without having to abandon any changes or additions which reflect requests from the CityGML community.

CityGML 2.0 is backwards compatible with version 1.0 in the following sense: each valid 1.0 instance is a valid 2.0 instance provided that the CityGML namespaces and schema locations in the document are changed to their actual 2.0 values. This step is required because the CityGML version number is encoded in these values, but no further actions have to be taken. Hence, there is a simple migration path from existing CityGML 1.0 instances to valid 2.0 instances.

The following clauses provide an overview of what is new in CityGML 2.0.

### New thematic modules for the representation of bridges and tunnels

Bridges and tunnels are important objects in city and landscape models. They are an essential part of the transportation infrastructure and are often easily recognizable landmarks of a city. CityGML 1.0 has been lacking thematic modules dedicated to bridges and tunnels, and thus such objects had to be modelled and exchanged using a GenericCityObject as proxy (cf. chapter 10.12). CityGML 2.0 now introduces two new thematic modules for the explicit representation of bridges and tunnels which complement the thematic model of CityGML: the Bridge module (cf. chapter 10.4) and the Tunnel module (cf. chapter 10.5).

Bridges and tunnels can be represented in LOD 1 – 4 and the underlying data models have a coherent structure with the Building model. For example, bridges and tunnels can be decomposed into parts, thematic boundary surfaces with openings are available to semantically classify parts of the shell, and installations as well as interi-
or built structures can be represented. This coherent model structure facilitates the similar understanding of semantic entities and helps to reduce software implementation efforts. Both the Bridge and the Tunnel model introduce further concepts and model elements which are specific to bridges and tunnels respectively.

**Additions to existing thematic modules**

- **CityGML Core module** (cf. chapter 10.1)
  Two new optional attributes have been added to the abstract base class `core:_CityObject` within the CityGML Core module: `relativeToTerrain` and `relativeToWater`. These attributes denote the feature’s location with respect to the terrain and water surface in a qualitative way, and thus facilitate simple and efficient queries (e.g., for the number of subsurface buildings) without the need for an additional digital terrain model or a model of the water body.

- **Building module** (cf. chapter 10.3)
  - **LOD0 representation**
    Buildings can now be represented in LOD0 by footprint and/or roof edge polygons. This allows the easy integration of existing 2D data and of roof reconstructions from aerial and satellite imagery into a 3D city model. The representations are restricted to horizontal, 3-dimensional surfaces.
  - **Additional thematic boundary surfaces**
    In order to semantically classify parts of the outer building shell which are neither horizontal wall surfaces nor parts of the roof, two additional boundary surfaces are introduced: `OuterFloorSurface` and `OuterCeilingSurface`.
  - **Additional relations to thematic boundary surfaces**
    In addition to `AbstractBuilding` and `Room`, the surface geometries of `BuildingInstallation` and `IntBuildingInstallation` features can now be semantically classified using thematic boundary surfaces. For example, this facilitates the semantic differentiation between roof and wall surfaces of dormers which are modeled as `BuildingInstallation`.
  - **Additional use of implicit geometries**
    Implicit geometries (cf. chapter 8.3) are now available for the representation of `Opening`, `BuildingInstallation`, and `IntBuildingInstallation` in addition to `BuildingFurniture`. A prototypical geometry for these city objects can thus be stored once and instantiated at different locations in the 3D city model.

- **Generics module** (cf. chapter 10.12)
  Two generic attributes have been added to the Generics module: `MeasureAttribute` and `GenericAttributeSet`. A `MeasureAttribute` facilitates the representation of measured values together with a reference to the employed unit. A `GenericAttributeSet` is a named collection of arbitrary generic attributes. It provides an optional `codeSpace` attribute to denote the authority organization who defined the attribute set.

- **LandUse module** (cf. chapter 10.10)
  The scope of the feature type `LandUse` has been broadened to comprise both areas of the earth’s surface dedicated to a specific land use and areas of the earth’s surface having a specific land cover with or without vegetation.

- **Attributes class, function, and usage (all modules)**
  In order to harmonize the use of the attributes `class`, `function`, and `usage`, this attribute triplet has been complemented for all feature classes that at least provided one of the attributes in CityGML 1.0.

**Additions to the CityGML code list mechanism**

In CityGML, code lists providing the allowed values for enumerative attributes such as `class`, `function`, and `usage` can be specified outside the CityGML schema by any organization or information community according to their specific information needs. This mechanism is, however, not fully reflected in the CityGML 1.0 encoding schema, because in a CityGML 1.0 instance document a corresponding attribute cannot point to the dictionary with the used code list values. This has been corrected for CityGML 2.0: All attributes taking values from code lists are now of type `gml:CodeType` following the GML 3.1.1 mechanism for the encoding of code list values (cf. ...
chapter 10.14 for more information). The gml:CodeType adds an optional codeSpace value to enumerative attributes which allows for providing a persistent URI pointing to the corresponding dictionary.

**Changelog for CityGML 2.0**

Changes on the level of XML schema components are provided in Annex F.

**Further edits to the specification document**

- **Accuracy requirements for Levels of Detail (LOD) (cf. chapter 6.2)**
  The accuracy requirements for the different CityGML LODs proposed in chapter 6.2 are non-normative. The wording of chapter 6.2 in CityGML 1.0 is however inconsistent with regard to this fact and thus has been clarified for CityGML 2.0.

- **Rework of the CityGML example datasets (cf. Annex G)**
  The CityGML examples provided in Annex G have been reworked and extended. They now show a consistent building model in all five LODs and demonstrate, for example, the semantic and geometric refinement of the building throughout the different LODs as well as the usage of XLinks to share geometry elements between features. The datasets are shipped with the CityGML XML Schema package, and are available at [http://schemas.opengis.net/citygml/examples/2.0/](http://schemas.opengis.net/citygml/examples/2.0/).

- **New example for the usage of Application Domain Extensions (cf. Annex I)**
  A second example for the usage of Application Domain Extensions in the field of Ubiquitous Network Robots Services has been added in Annex I.
OGC® City Geography Markup Language (CityGML)
Encoding Standard

1 Scope

This document is an OGC Encoding Standard for the representation, storage and exchange of virtual 3D city and landscape models. CityGML is implemented as an application schema of the Geography Markup Language version 3.1.1 (GML3).

CityGML models both complex and georeferenced 3D vector data along with the semantics associated with the data. In contrast to other 3D vector formats, CityGML is based on a rich, general purpose information model in addition to geometry and appearance information. For specific domain areas, CityGML also provides an extension mechanism to enrich the data with identifiable features under preservation of semantic interoperability.

Targeted application areas explicitly include urban and landscape planning; architectural design; tourist and leisure activities; 3D cadastres; environmental simulations; mobile telecommunications; disaster management; homeland security; vehicle and pedestrian navigation; training simulators and mobile robotics.

CityGML is considered a source format for 3D portraying. The semantic information contained in the model can be used in the styling process which generates computer graphics represented e.g. as KML/COLLADA or X3D files. The appropriate OGC Portrayal Web Service for this process is the OGC Web 3D Service (W3DS). An image-based 3D portrayal service for virtual 3D landscape and city models is provided by the OGC Web View Service (WVS).

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
  - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
  - Sites (currently buildings, bridges, and tunnels)
  - Vegetation (areas, volumes, and solitary objects with vegetation classification)
  - Water bodies (volumes, surfaces)
  - Transportation facilities (both graph structures and 3D surface data)
  - Land use (representation of areas of the earth’s surface dedicated to a specific land use)
  - City furniture
  - Generic city objects and attributes
  - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
  - LOD0 – regional, landscape
  - LOD1 – city, region
  - LOD2 – city districts, projects
  - LOD3 – architectural models (outside), landmarks
  - LOD4 – architectural models (interior)
- Multiple representations in different LODs simultaneously; generalisation relations between objects in different LODs
- Optional topological connections between feature (sub)geometries
- Application Domain Extensions (ADE): Specific “hooks” in the CityGML schema allow to define application specific extensions, for example for noise pollution simulation, or to augment CityGML by properties of the new National Building Information Model Standard (NBIMS) in the U.S.
2 Conformance

Conformance targets addressed by this International standard are CityGML instance documents only. Future revisions of this International Standard may also address consumers or producers as conformance targets.

Clauses 8 to 10 of this International standard specify separate CityGML XML Schema definitions and normative aspects, i.e. CityGML modules, which shall be used in CityGML instance documents in accordance with clause 7. Implementations are not required to support the full range of capabilities provided by the universe of all CityGML modules. Valid partial implementations are supported following the rules and guidelines for CityGML profiles in chapter 7.2.

CityGML instance documents claiming conformance to this International Standard shall:

a) conform to the rules and requirements specified in clauses 7 to 10;

b) pass all relevant test cases of the abstract test suite in annex B.1;

c) satisfy all relevant conformance classes of the abstract test suite related to CityGML modules in annex B.2.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of OGC 12-019. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-019 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

The following documents are indispensable for the application of the CityGML standard. The geometry model of GML 3.1.1 is used except for some added concepts like implicit geometries (cf. chapter 8.2). The appearance model (cf. chapter 9) draws concepts from both X3D and COLLADA. Addresses are represented using the OASIS extensible Address Language xAL.

ISO 8601:2004, 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
OpenGIS® Abstract Specification Topic 0, Overview, OGC document 04-084
OpenGIS® Abstract Specification Topic 8, Relations between Features, OGC document 99-108r2
OpenGIS® Geography Markup Language Implementation Specification, Version 3.1.1, OGC document 03-105r1
OpenGIS® GML 3.1.1 Simple Dictionary Profile, Version 1.0.0, OGC document 05-099r2
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).
Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.4.1
The Schematron Assertion Language 1.5. Rick Jelliffe 2002-10-01

4 Conventions

4.1 Abbreviated terms

The following abbreviated terms are used in this document:
2D Two Dimensional
3D Three Dimensional
AEC Architecture, Engineering, Construction
ALKIS German National Standard for Cadastral Information
ATKIS German National Standard for Topographic and Cartographic Information
B-Rep Boundary Representation
bSI buildingSMART International
CAD Computer Aided Design
COLLADA Collaborative Design Activity
CSG Constructive Solid Geometry
DTM Digital Terrain Model
DXF Drawing Exchange Format
EuroSDR European Spatial Data Research Organisation
ESRI Environmental Systems Research Institute
FM Facility Management
GDF Geographic Data Files
GDI-DE Spatial Data Infrastructure Germany (Geodateninfrastruktur Deutschland)
GDI NRW Geodata Infrastructure North-Rhine Westphalia
GML Geography Markup Language
IAI International Alliance for Interoperability (now buildingSMART International (bSI))
IETF Internet Engineering Task Force
IFC Industry Foundation Classes
ISO International Organization for Standardisation
LOD Level of Detail
NBIMS National Building Information Model Standard
OASIS Organisation for the Advancement of Structured Information Standards
OGC Open Geospatial Consortium
OSCRE Open Standards Consortium for Real Estate
4.2 UML Notation

The CityGML standard 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).

According to GML3 all associations between model elements in CityGML are unidirectional. Thus, associations in CityGML are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.
The following stereotypes are used:

<<Geometry>> represents the geometry of an object. The geometry is an identifiable and distinguishable object that is derived from the abstract GML type AbstractGeometryType.

<<Feature>> represents a thematic feature according to the definition in ISO 19109. A feature is an identifiable and distinguishable object that is derived from the abstract GML type AbstractFeatureType.

<<Object>> represents an identifiable and distinguishable object that is derived from the abstract GML type AbstractGMLType.

<<Enumeration>> enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified inline the CityGML schema.

<<CodeList>> enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline the CityGML schema. The allowed values can be provided within an external code list. It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. chapter 6.6 and chapter 10.14).

<<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.

<<PrimitiveType>> is used for representations supported by a primitive type in the implementation.

<<DataType>> is used as a descriptor of a set of values that lack identity. Data types include primitive predefined types and user-definable types. A DataType is thus a class with few or no operations whose primary purpose is to hold the abstract state of another class for transmittal, storage, encoding or persistent storage.

<<Leaf>> is used within UML package diagrams to indicate model elements that can have no further subtypes.

<<XSDSchema>> is used within UML package diagrams to denote the root element of an XSD Schema containing all the definitions for a particular namespace. All the package contents or component classes are placed within the one schema.

<<ApplicationSchema>> is used within UML package diagrams to denote an XML Schema definition fundamentally dependent on the concepts of another independent Standard within the XML Schema metala nguage. For example, ApplicationSchema indicates extensions of GML consistent with the GML “rules for application schemas”.

In order to enhance the readability of the CityGML UML diagrams, classes are depicted in different colors if they belong to different UML packages (see Fig. 8 for an overview of UML packages). The following coloring scheme is applied:

- **Classes painted in yellow** belong to the UML package which is subject of discussion in that clause of the specification in which the UML diagram is given. For example, in the context of chapter 10.1 which introduces the CityGML Core module, the yellow color is used to denote classes which are defined in the CityGML Core UML package. Likewise, the yellow classes shown in UML diagrams in chapter 10.3 are associated with the Building module which is subject of discussion in that chapter.

- **Classes painted in blue** belong to a CityGML UML package different to that associated with the yellow color. In order to explicitly denote the UML package of such classes, their class names carry a namespace prefix which is uniquely associated with a CityGML module throughout this specification (cf. section 4.3 for a list of namespaces and prefixes). For example, in the context of the Building module, classes from the CityGML Core module are painted in blue and their class names are preceded by the prefix core.

- **Classes painted in green** are defined in GML3 and their class names are preceded by the prefix gml.
The following example UML diagram demonstrates the UML notation and coloring scheme used throughout this specification. In this example, the yellow classes are associated with the CityGML Building module, the blue classes are from the CityGML Core module, and the green class depicts a geometry element defined by GML3.

Fig. 2: Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

### 4.3 XML namespaces and namespace prefixes

The CityGML data model is thematically decomposed into a core module and thematic extension modules. All modules including the core are specified by their own XML schema file, each defining a globally unique XML namespace. The extension modules are based on the core module and, thus, contain (by reference) the CityGML core schema.

Within this document the module namespaces are associated with recommended prefixes. These prefixes are consistently used within the normative parts of this specification, for all UML diagrams and example CityGML instance documents. The CityGML core and extension modules along with their XML namespace identifiers and recommended namespace prefixes are listed in Tab. 1.

<table>
<thead>
<tr>
<th>CityGML module</th>
<th>Namespace identifier</th>
<th>Namespace prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityGML Core</td>
<td><a href="http://www.opengis.net/citygml/2.0">http://www.opengis.net/citygml/2.0</a></td>
<td>core</td>
</tr>
<tr>
<td>Appearance</td>
<td><a href="http://www.opengis.net/citygml/appearance/2.0">http://www.opengis.net/citygml/appearance/2.0</a></td>
<td>app</td>
</tr>
<tr>
<td>Bridge</td>
<td><a href="http://www.opengis.net/citygml/bridge/2.0">http://www.opengis.net/citygml/bridge/2.0</a></td>
<td>brid</td>
</tr>
<tr>
<td>Building</td>
<td><a href="http://www.opengis.net/citygml/building/2.0">http://www.opengis.net/citygml/building/2.0</a></td>
<td>bldg</td>
</tr>
<tr>
<td>CityFurniture</td>
<td><a href="http://www.opengis.net/citygml/cityfurniture/2.0">http://www.opengis.net/citygml/cityfurniture/2.0</a></td>
<td>frn</td>
</tr>
<tr>
<td>CityObjectGroup</td>
<td><a href="http://www.opengis.net/citygml/cityobjectgroup/2.0">http://www.opengis.net/citygml/cityobjectgroup/2.0</a></td>
<td>grp</td>
</tr>
<tr>
<td>Generics</td>
<td><a href="http://www.opengis.net/citygml/generics/2.0">http://www.opengis.net/citygml/generics/2.0</a></td>
<td>gen</td>
</tr>
<tr>
<td>LandUse</td>
<td><a href="http://www.opengis.net/citygml/landuse/2.0">http://www.opengis.net/citygml/landuse/2.0</a></td>
<td>luse</td>
</tr>
<tr>
<td>Relief</td>
<td><a href="http://www.opengis.net/citygml/relief/2.0">http://www.opengis.net/citygml/relief/2.0</a></td>
<td>dem</td>
</tr>
<tr>
<td>Transportation</td>
<td><a href="http://www.opengis.net/citygml/transportation/2.0">http://www.opengis.net/citygml/transportation/2.0</a></td>
<td>tran</td>
</tr>
<tr>
<td>Tunnel</td>
<td><a href="http://www.opengis.net/citygml/tunnel/2.0">http://www.opengis.net/citygml/tunnel/2.0</a></td>
<td>tun</td>
</tr>
<tr>
<td>Vegetation</td>
<td><a href="http://www.opengis.net/citygml/vegetation/2.0">http://www.opengis.net/citygml/vegetation/2.0</a></td>
<td>veg</td>
</tr>
<tr>
<td>WaterBody</td>
<td><a href="http://www.opengis.net/citygml/waterbody/2.0">http://www.opengis.net/citygml/waterbody/2.0</a></td>
<td>wtr</td>
</tr>
<tr>
<td>TexturedSurface [deprecated]</td>
<td><a href="http://www.opengis.net/citygml/texturedsurface/2.0">http://www.opengis.net/citygml/texturedsurface/2.0</a></td>
<td>tex</td>
</tr>
</tbody>
</table>

Tab. 1: List of CityGML modules, their associated XML namespace identifiers, and example namespace prefixes.
Further XML Schema definitions relevant to this standard are shown in Tab. 2 along with the corresponding XML namespace identifiers and namespace prefixes consistently used within this document.

<table>
<thead>
<tr>
<th>XML Schema definition</th>
<th>Namespace identifier</th>
<th>Namespace prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geography Markup Language version 3.1.1 (from OGC)</td>
<td><a href="http://www.opengis.net/gml">http://www.opengis.net/gml</a></td>
<td>gml</td>
</tr>
<tr>
<td>Extensible Address Language version 2.0 (from OASIS)</td>
<td>urn:oasis:names:tc:ciq:xsd:AL:2.0</td>
<td>xAL</td>
</tr>
<tr>
<td>Schematron Assertion Language version 1.5</td>
<td><a href="http://www.ascc.net/xml/schematron">http://www.ascc.net/xml/schematron</a></td>
<td>sch</td>
</tr>
</tbody>
</table>

Tab. 2: List of XML Schema definitions, their associated XML namespace identifiers, and example namespace prefixes used within this document.

### 4.4 XML-Schema

The normative parts of the standard 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 standard is not intended to serve as an introduction to XML schema. In order to have a full understanding of this candidate standard, it is necessary for the reader to have a reasonable knowledge of XML schema.
5 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 version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

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

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 which is a subclass of the GML class _Feature. 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, bridges, and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), land use, water bodies, transportation facilities, and city furniture (chapter 10). Further objects, which are not explicitly modelled yet, can be represented using the concept of generic objects and attributes (chapter 6.11). In addition, extensions to the CityGML data model applying to specific application fields can be realised using the Application Domain Extensions (ADE) (chapter 6.12). 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 8.2). A grouping concept allows the combination of single 3D objects, e.g. buildings to a building complex (chapter 6.8). Objects which are not geometrically modelled by closed solids can be virtually sealed in order to compute their volume (e.g. pedestrian underpasses, tunnels, or airplane hangars). They can be closed using ClosureSurfaces (chapter 6.4). 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 6.5).

CityGML differentiates five consecutive Levels of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (chapter 6.2). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Generalisation relations allow the explicit representation of aggregated objects over different scales.

In addition to spatial properties, CityGML features can be assigned appearances. Appearances are not limited to visual data but represent arbitrary observable properties of the feature’s surface such as infrared radiation, noise pollution, or earthquake-induced structural stress (chapter 9).

Furthermore, objects can have external references to corresponding objects in external datasets (chapter 6.7). The possible attribute values of enumerative object attributes can be enumerated in code lists defined in external, redefinable dictionaries (chapter 6.6).
6 General characteristics of CityGML

6.1 Modularisation

The CityGML data model consists of class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. However, implementations are not required to support the overall CityGML data model in order to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. For this purpose, modularisation is applied to the CityGML data model (cf. chapter 7).

The CityGML data model is thematically decomposed into a core module and thematic extension modules. The core module comprises the basic concepts and components of the CityGML data model and, thus, must be implemented by any conformant system. Based on the core module, each extension covers a specific thematic field of virtual 3D city models. CityGML introduces the following thirteen thematic extension modules: Appearance, Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody, and TexturedSurface [deprecated].

CityGML compliant implementations may support any combination of extension modules in conjunction with the core module. Such combinations of modules are called CityGML profiles. Therefore, CityGML profiles allow for valid partial implementations of the overall CityGML data model.

6.2 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 visualisation and data analysis (see Fig. 3). In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualisation 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. However, it will be within the responsibility of the user or application to make sure objects in different LODs refer to the same real-world object.

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. Buildings may be represented in LOD0 by footprint or roof edge polygons. LOD1 is the well-known blocks model comprising prismatic buildings with flat roof structures. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated boundary surfaces. LOD3 denotes architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes a LOD3 model by adding interior structures for buildings. For example, buildings in LOD4 are composed of rooms, interior doors, stairs, and furniture. In all LODs appearance information such as high-resolution textures can be mapped onto the structures (cf. 6.9).

![LOD0, LOD1, LOD2, LOD3, LOD4](Fig. 3: The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn))
LODs are also characterised by differing accuracies and minimal dimensions of objects (cf. Tab. 3). The accuracy requirements given in this standard are debatable and are to 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 and it is typically much higher than the absolute accuracy. In LOD1, the positional and height accuracy of points should be 5m or less, while all objects with a footprint of at least 6m by 6m should be considered. The positional and height accuracy of LOD2 is proposed to be 2m or better. In this LOD, all objects with a footprint of at least 4m \( \times \) 4m should be considered. Both types of accuracies in LOD3 should be 0.5m, and the minimal footprint is suggested to be 2m \( \times \) 2m. Finally, the positional and height accuracy of LOD4 should 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 categorisation 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>Class of accuracy</td>
<td>lowest</td>
<td>low</td>
<td>middle</td>
<td>high</td>
<td>very high</td>
</tr>
<tr>
<td>Absolute 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>0/0.2m</td>
</tr>
<tr>
<td>Generalisation</td>
<td>maximal generalisation</td>
<td>object blocks as generalised features; &gt; 6*6m/3m</td>
<td>objects as generalised features; &gt; 4*4m/2m</td>
<td>object as real features; &gt; 2*2m/1m</td>
<td>constructive elements and openings are represented</td>
</tr>
<tr>
<td>Building installations</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>representative exterior features</td>
<td>real object form</td>
</tr>
<tr>
<td>Roof structure/representation</td>
<td>yes</td>
<td>flat</td>
<td>differentiated roof structures</td>
<td>real object form</td>
<td>real object form</td>
</tr>
<tr>
<td>Roof overhanging parts</td>
<td>yes</td>
<td>no</td>
<td>yes, if known</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CityFurniture</td>
<td>no</td>
<td>important objects</td>
<td>prototypes, generalised objects</td>
<td>real object form</td>
<td>real object form</td>
</tr>
<tr>
<td>SolitaryVegetationObject</td>
<td>no</td>
<td>important objects</td>
<td>prototypes, higher 6m</td>
<td>prototypes, higher 2m</td>
<td>prototypes, real object form</td>
</tr>
<tr>
<td>PlantCover</td>
<td>no</td>
<td>&gt;50*50m</td>
<td>&gt;5*5m</td>
<td>&lt; LOD2</td>
<td>&lt; LOD2</td>
</tr>
</tbody>
</table>

Tab. 3: LOD 0-4 of CityGML with their proposed accuracy requirements (discussion proposal, based on: Albert et al. 2003).

Whereas in CityGML each object can have a different representation for every LOD, often different objects from the same LOD will be generalised to be represented by an aggregate object in a lower LOD. CityGML supports the aggregation / decomposition by providing an explicit generalisation association between city objects (further details see UML diagram in chapter 10.1).

### 6.3 Coherent semantical-geometrical modelling

One of the most important design principles for CityGML is the coherent modelling 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 (cf. Stadler & Kolbe 2007). 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 ensured 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.

### 6.4 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 a **ClosureSurface**. These 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 visualisations.

The concept of ClosureSurface 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 visualisations the entrance must be treated as open. Thus, closure surfaces 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: IGG Uni Bonn).

### 6.5 Terrain Intersection Curve (TIC)

A crucial issue in city modelling is the integration of 3D objects and the terrain. Problems arise if 3D objects float over or sink into the terrain. This is particularly the case if terrains and 3D objects in different LOD are combined, or if they come from different providers (Kolbe and Gröger 2003). To overcome this problem, the TerrainIntersectionCurve (TIC) of a 3D object is introduced. These curves denote the exact position, where the terrain touches the 3D object (see Fig. 5). TICs can be applied to buildings and building parts (cf. chapter 10.3), bridge, bridge parts and bridge construction elements (cf. chapter 10.5), tunnel and tunnel parts (cf. chapter 10.4), city furniture objects (cf. chapter 10.9), and generic city objects (cf. chapter 10.12). If, for example, a 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 3D object 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: IGG Uni Bonn).
6.6 Code lists for enumerative attributes

CityGML feature types often include attributes whose values can be enumerated in a list of discrete values. An example is the attribute *roof type* of a building, 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. Moreover, the list of possible attribute values often is not fixed and may substantially vary for different countries (e.g., due to national law and regulations) and for different information communities.

In CityGML, such enumerative attributes are of type `gml:CodeType` and their allowed attribute values can be provided in a *code list* which is specified outside the CityGML schema. A code list contains coded attribute values and ensures that the same code is used for the same notion or concept. If a code list is provided for an enumerative attribute, the attribute may only take values from this list. This allows applications to validate the attribute value and thus facilitates semantic and syntactic interoperability. It is recommended that code lists are implemented as *simple dictionaries* following the GML 3.1.1 Simple Dictionary Profile (cf. Whiteside 2005).

The governance of code lists is decoupled from the governance of the CityGML schema and specification. Thus, code lists may be specified by any organisation or information community according to their information needs. There shall be one authority per code list who is in charge of the code list values and the maintenance of the code list. Further information on the CityGML code list mechanism is provided in chapter 10.14.

Code lists can have references to existing models. For example, room codes defined by the Open Standards Consortium for Real Estate (OSCRE) can be referenced or classifications of buildings and building parts introduced by the National Building Information Model Standard (NBIMS) can be used. Annex C contains non-normative code lists proposed by the SIG 3D for almost all enumerative attributes in CityGML. They can be directly referenced in CityGML instance documents and serve as an example for the definition of code lists.

6.7 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 refer to external data sets (for the UML diagram see Fig. 21; and for XML schema definition see annex A.1) using the concept of `ExternalReference`. 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 on the internet. The generic concept of external references allows for any _CityObject an arbitrary number of links to corresponding objects in external information systems (e.g. ALKIS, ATKIS, OS MasterMap®, GDF, etc.).

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

6.8 City object groups

The grouping concept of CityGML allows for 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 10.11; XML schema definition see annex A.6). 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, for example, 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. The grouping concept is delivered by the thematic extension module CityObjectGroup of CityGML (cf. chapter 10.11).

6.9 Appearances

Information about a surface’s appearance, i.e. observable properties of the surface, is considered an integral part of virtual 3D city models in addition to semantics and geometry. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties. Consequently, data provided by appearances can be used as input for both presentation of and analysis in virtual 3D city models.

CityGML supports feature appearances for an arbitrary number of themes per city model. Each LOD of a feature can have an individual appearance. Appearances can represent – among others – textures and georeferenced textures. CityGML’s appearance model is packaged within its own extension module Appearance (cf. chapter 9).

6.10 Prototypic objects / scene graph concepts

In CityGML, objects of equal shape like trees and other vegetation objects, traffic lights and 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 (for further description cf. chapter 8.2).

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

6.11 Generic city objects and attributes

CityGML is being designed as a universal topographic information model that defines object types and attributes which are useful for a broad range of applications. 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. CityGML provides two different
concepts to support the exchange of such data: 1) generic objects and attributes, and 2) Application Domain Extensions (cf. chapter 6.12).

The concept of generic objects and attributes allows for the extension of CityGML applications during runtime, i.e. any _CityObject may be augmented by additional attributes, whose names, data types, and values can be provided by a running application without any change of the CityGML XML schema. Similarly, features not represented by the predefined thematic classes of the CityGML data model may be modelled and exchanged using generic objects. The generic extensions of CityGML are provided by the thematic extension module Generics (cf. chapter 10.12).

The current version of CityGML does not include, for example, explicit thematic models for embankments, excavations and city walls. These objects may be stored or exchanged using generic objects and attributes.

6.12 Application Domain Extensions (ADE)

Application Domain Extensions (ADE) specify additions to the CityGML data model. Such additions comprise the introduction of new properties to existing CityGML classes like e.g. the number of habitants of a building or the definition of new object types. The difference between ADEs and generic objects and attributes is, that an ADE has to be defined in an extra XML schema definition file with its own namespace. This file has to explicitly import the XML Schema definition of the extended CityGML modules.

The advantage of this approach is that the extension is formally specified. Extended CityGML instance documents can be validated against the CityGML and the respective ADE schema. ADEs can be defined (and even standardised) by information communities which are interested in specific application fields. More than one ADE can be actively used in the same dataset (further description cf. chapter 10.13).

ADEs may be defined for one or even several CityGML modules providing a high flexibility in adding additional information to the CityGML data model. Thus, the ADE mechanism is orthogonally aligned with the modularisation approach of CityGML. Consequently, there is no separate extension module for ADEs.

In this specification, two examples for ADEs are included:

- An ADE for Noise Immission Simulation (Annex H) which is employed in the simulation of environmental noise dispersion according to the Environmental Noise Directive of the European Commission (2002/49/EC);
- An ADE for Ubiquitous Network Robots Services (Annex I) which demonstrates the usage of CityGML for the navigation of robots in indoor environments.

Further examples for ADEs are the CAFM ADE (Bleifuß et al., 2009) for facility management, the UtilityNetworkADE (Becker et al., 2011) for the integrated 3D modeling of multi-utility networks and their interdependencies, the HydroADE (Schulte and Coors, 2008) for hydrographical applications and the GeoBIM (IFC) ADE (van Berlo et al., 2011) which combines BIM information from IFC (from bSI) with CityGML and is implemented in the open source modelserver BIMserver.org.
7 Modularisation

CityGML is a rich standard both on the thematic and geometric-topological level of its data model. On its thematic level CityGML defines classes and relations for the most relevant topographic objects in cities and regional models comprising built structures, elevation, vegetation, water bodies, city furniture, and more. In addition to geometry and appearance content these thematic components allow to employ virtual 3D city models for sophisticated analysis tasks in different application domains like simulations, urban data mining, facility management, and thematic inquiries.

CityGML is to be seen as a framework giving geospatial 3D data enough space to grow in geometrical, topological and semantic aspects over its lifetime. Thus, geometry and semantics of city objects may be flexibly structured covering purely geometric datasets up to complex geometric-topologically sound and spatio-semantically coherent data. By this means, CityGML defines a single object model and data exchange format applicable to consecutive process steps of 3D city modelling from geometry acquisition, data qualification and refinement to preparation of data for specific end-user applications, allowing for iterative data enrichment and lossless information exchange (cf. Kolbe et al. 2009).

According to this idea of a framework, applications are not required to support all thematic fields of CityGML in order to be compliant to the standard, but may employ a subset of constructs corresponding to specific relevant requirements of an application domain or process step. The use of logical subsets of CityGML limits the complexity of the overall data model and explicitly allows for valid partial implementations. As for version 2.0 of the CityGML standard, possible subsets of the data model are defined and embraced by so called CityGML modules. A CityGML module is an aggregate of normative aspects that must all be implemented as a whole by a conformant system. CityGML consists of a core module and thematic extension modules.

The CityGML core module defines the basic concepts and components of the CityGML data model. It is to be seen as the universal lower bound of the overall CityGML data model and a dependency of all thematic extension modules. Thus, the core module is unique and must be implemented by any conformant system. Based on the CityGML core module, each extension module contains a logically separate thematic component of the CityGML data model. The extensions to the core are derived by vertically slicing the overall CityGML data model. Since the core module is contained (by reference) in each extension module, its general concepts and components are universal to all extension modules. The following thirteen thematic extension modules are introduced by version 2.0 of the CityGML standard. They are directly related to clauses of this document each covering the corresponding thematic field of CityGML:

- Appearance (cf. clause 9),
- Bridge (cf. clause 10.5),
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.9),
- CityObjectGroup (cf. clause 10.11),
- Generics (cf. clause 10.12),
- LandUse (cf. clause 10.10),
- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.7),
- Tunnel (cf. clause 10.4),
- Vegetation (cf. clause 10.8),
- WaterBody (cf. clause 10.6), and
- TexturedSurface [deprecated] (cf. clause 9.8).

The thematic decomposition of the CityGML data model allows for implementations to support any combination of extension modules in conjunction with the core module in order to be CityGML conformant. Thus, the extension modules may be arbitrarily combined according to the information needs of an application or application domain. A combination of modules is called a CityGML profile. The union of all modules is defined as the
CityGML base profile. The base profile is unique at any given time and forms the upper bound of the overall CityGML data model. Any other CityGML profile must be a valid subset of the base profile. By following the concept of CityGML modules and profiles, valid partial implementations of the CityGML data model may be realised in a well-defined way.

As for future development, each CityGML module may be further developed independently from other modules by expert groups and information communities. Resulting proposals and changes to modules may be introduced into future revisions of the CityGML standard without affecting the validity of other modules. Furthermore, thematic components not covered by the current CityGML data model may be added to future revisions of the standard by additional thematic extension modules. These additional extensions may establish dependency relations to any other existing CityGML module but shall at least be dependent on the CityGML core module. Consequently, the CityGML base profile may vary over time as new extensions are added. However, if a specific application has information needs to be modelled and exchanged which are beyond the scope of the CityGML data model, this application data can also be incorporated within the existing modules using CityGML’s Application Domain Extension mechanism (cf. clause 10.13) or by employing the concepts of generic city objects and attributes (cf. chapter 10.12).

The introduced modularisation approach supports CityGML’s versatility as a data modelling framework and exchange format addressing various application domains and different steps of 3D city modelling. For sake of clarity, applications should announce the level of conformance to the CityGML standard by declaring the employed CityGML profile. Since the core module is part of all profiles, this should be realised by enumerating the implemented thematic extension modules. For example, if an implementation supports the Building module, the Relief module, and the Vegetation module in addition to the core, this should be announced by “CityGML [Building, Relief, Vegetation]”. In case the base profile is supported, this should be indicated by “CityGML [full]”.

7.1 CityGML core and extension modules

Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, a single namespace shall not be directly included in two modules. Thus, all elements belonging to one module are associated to the module’s namespace only. By this means, module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

Compared to CityGML versions before 1.0, the aforementioned namespace conventions introduce an extra level of complexity to data files as there is no single CityGML namespace any more. In contrast, components of different CityGML modules and, thus, of different namespaces may be arbitrarily mixed within the same CityGML instance document. Furthermore, an application might have to parse instance documents containing elements of modules which are not employed by the application itself. These parsing problems though can easily be overcome by non-“schema-aware” applications, i.e. applications that do not parse and interpret GML application schemas in a generic way. Elements from different namespaces than those declared by the application’s employed CityGML profile could be skipped. Comparable observations have to be made when using CityGML’s Application Domain Extension mechanism (cf. clause 10.13).

As for version 2.0 of the CityGML standard, there are no two thematic extension modules related by dependency. Thus, all extension modules are truly independent from each other and may be separately supported by implementations. However, the CityGML core module is a dependency for any extension module. This means that the XML schema file of the core module is imported by each XML schema file defining an extension.

The dependency relations between CityGML’s modules are illustrated in Fig. 8 using an UML package diagram. Each module is represented by a package. The package names correspond to the module names. A dashed arrow in the figure indicates that the schema at the tail of the arrow depends upon the schema at the head of the arrow. For CityGML modules, a dependency occurs where one schema <import>s another schema and accordingly the corresponding XML namespace. For example, the extension module Building imports the schema of the CityGML Core module. A short description of each module is given in Tab. 4.
Fig. 8: UML package diagram illustrating the separate modules of CityGML and their schema dependencies. Each extension module (indicated by the leaf packages) further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes. For readability reasons, the corresponding dependencies have been omitted.

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityGML Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/2.0">http://www.opengis.net/citygml/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityGMLBase.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>core</td>
</tr>
<tr>
<td>Module description</td>
<td>The CityGML Core module defines the basic components of the CityGML data model. Primarily, this comprises abstract base classes from which all thematic classes are (transitively) derived. But also non-abstract content common to more than one extension module, for example basic data types, is defined within the core module. The core module itself imports the XML schema definition files of GML version 3.1.1 and the OASIS extensible Address Language xAL.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/appearance/2.0">http://www.opengis.net/citygml/appearance/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>appearance.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>app</td>
</tr>
<tr>
<td>Module description</td>
<td>The Appearance module provides the means to model appearances of CityGML features, i.e. observable properties of the feature’s surface. Appearance data may be stored for each city object. Therefore, the abstract base class _CityObject defined within the core module is augmented by an additional property using CityGML’s Application Domain Extension mechanism. Thus, the Appearance module has a deliberate impact on all thematic extension modules.</td>
</tr>
<tr>
<td>Module name</td>
<td>Bridge</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/bridge/2.0">http://www.opengis.net/citygml/bridge/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>bridge.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>brid</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Bridge</em> module allows the representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures in four levels of detail (LOD 1 – 4).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/building/2.0">http://www.opengis.net/citygml/building/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>building.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>bldg</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Building</em> module allows the representation of thematic and spatial aspects of buildings, building parts, building installations, and interior building structures in five levels of detail (LOD 0 – 4).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityFurniture</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/cityfurniture/2.0">http://www.opengis.net/citygml/cityfurniture/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityFurniture.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>fn</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>CityFurniture</em> module is used to represent city furniture objects in cities. City furniture objects are immovable objects like lanterns, traffic signs, advertising columns, benches, or bus stops that can be found in traffic areas, residential areas, on squares, or in built-up areas.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>CityObjectGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/cityobjectgroup/2.0">http://www.opengis.net/citygml/cityobjectgroup/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>cityObjectGroup.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>grp</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>CityObjectGroup</em> module provides a grouping concept for CityGML. Arbitrary city objects may be aggregated in groups according to user-defined criteria to represent and transfer these aggregations as part of the city model. A group may be further classified by specific attributes.</td>
</tr>
<tr>
<td>Module name</td>
<td>Generics</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/generics/2.0">http://www.opengis.net/citygml/generics/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>generics.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>gen</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Generics</em> module provides generic extensions to the CityGML data model that may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. However, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module. In order to represent generic attributes, the <em>Generics</em> module augments the abstract base class <code>_CityObject</code> defined within the core module by an additional property using CityGML’s <em>Application Domain Extension</em> mechanism. Thus, the <em>Generics</em> module has a deliberate impact on all thematic extension modules.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>LandUse</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/landuse/2.0">http://www.opengis.net/citygml/landuse/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>landUse.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>luse</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>LandUse</em> module allows for the representation of areas of the earth’s surface dedicated to a specific land use.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Relief</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/relief/2.0">http://www.opengis.net/citygml/relief/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>relief.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>dem</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Relief</em> module allows for the representation of the terrain in a city model. CityGML supports terrain representations in different levels of detail, reflecting different accuracies or resolutions. The terrain may be specified as a regular raster or grid, as a TIN, by break lines, and by mass points.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Transportation</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/transportation/2.0">http://www.opengis.net/citygml/transportation/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>transportation.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>tran</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Transportation</em> module is used to represent the transportation features within a city, for example roads, tracks, railways, or squares. Transportation features may be represented as a linear network or by geometrically describing their 3D surfaces.</td>
</tr>
<tr>
<td>Module Name</td>
<td>Tunnel</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/tunnel/2.0">http://www.opengis.net/citygml/tunnel/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>tunnel.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>tun</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Tunnel</em> module facilitates the representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures in four level of detail (LOD 1 – 4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>Vegetation</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/vegetation/2.0">http://www.opengis.net/citygml/vegetation/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>vegetation.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>veg</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>Vegetation</em> module provides thematic classes to represent vegetation objects. CityGML’s vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>WaterBody</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/waterbody/2.0">http://www.opengis.net/citygml/waterbody/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>waterBody.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>wtr</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>WaterBody</em> module represents the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects so far.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module name</th>
<th>TexturedSurface [deprecated]</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML namespace identifier</td>
<td><a href="http://www.opengis.net/citygml/texturedsurface/2.0">http://www.opengis.net/citygml/texturedsurface/2.0</a></td>
</tr>
<tr>
<td>XML Schema file</td>
<td>texturedSurface.xsd</td>
</tr>
<tr>
<td>Recommended namespace prefix</td>
<td>tex</td>
</tr>
<tr>
<td>Module description</td>
<td>The <em>TexturedSurface</em> module allows for assigning visual appearance properties (color, shininess, transparency) and textures to 3D surfaces. Due to inherent limitations of its modelling approach this module has been marked deprecated and is expected to be removed in future CityGML versions. Appearance information provided by this module can be converted to CityGML’s <em>Appearance</em> module without information loss. Thus, the use of the <em>TexturedSurface</em> module is strongly discouraged.</td>
</tr>
</tbody>
</table>

Tab. 4: Overview of CityGML’s core and thematic extensions modules.
7.2 CityGML profiles

A CityGML profile is a combination of thematic extension modules in conjunction with the core module of CityGML. Each CityGML instance document shall employ the CityGML profile appropriate to the provided data. In general, two approaches to employ a CityGML profile within an instance document can be differentiated:

1. **CityGML profile definition embedded inline the CityGML instance document**

A CityGML profile can be bound to an instance document using the `schemaLocation` attribute defined in the XML Schema instance namespace, `http://www.w3.org/2001/XMLSchema-instance` (commonly associated with the prefix `xs`). The `xs:schemaLocation` attribute provides a way to locate the XML Schema definition for namespaces defined in an XML instance document. Its value is a whitespace-delimited list of pairs of Uniform Resource Identifiers (URIs) where each pair consists of a namespace followed by the location of that namespace’s XML Schema definition, which is typically a .xsd file.

By this means, the namespaces of the respective CityGML modules shall be defined within a CityGML instance document. The `xs:schemaLocation` attribute then shall be used to provide the location to the respective XML Schema definition of each module. All example instance documents given in Annex G follow this first approach.

2. **CityGML profile definition provided by a separate XML Schema definition file**

The CityGML profile may also be specified by its own XML Schema file. This schema file shall combine the appropriate CityGML modules by importing the corresponding XML Schema definitions. For this purpose, the `import` element defined in the XML Schema namespace shall be used, `http://www.w3.org/2001/XMLSchema-instance` (commonly associated with the prefix `xs`). For the `xs:import` element, the namespace of the imported CityGML module along with the location of the namespace’s XML Schema definition have to be declared. In order to apply a CityGML profile to an instance document, the profile’s schema has to be bound to the instance document using the `xs:schemaLocation` attribute. The XML Schema file of the CityGML profile shall not contain any further content.

The `targetNamespace` of the profile’s schema shall differ from the namespaces of the imported CityGML modules. The namespace associated with the profile should be in control of the originator of the instance document and must be given as a previously unused and globally unique URI. The profile’s XML Schema file must be available (or accessible on the internet) to everybody parsing the associated CityGML instance document.

The second approach is illustrated by the following example XML Schema definition for the base profile of CityGML. Since the base profile is the union of all CityGML modules, the corresponding XML Schema definition imports each and every CityGML module. By this means, all components of the CityGML data model are available in and may be exchanged by instance documents referencing this example base profile. The schema definition file of the base profile is shipped with the CityGML schema package, and is accessible at [http://schemas.opengis.net/citygml/profiles/base/2.0/CityGML.xsd](http://schemas.opengis.net/citygml/profiles/base/2.0/CityGML.xsd).

```xml
  <xs:import namespace="http://www.opengis.net/citygml/appearance/2.0" schemaLocation="http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/bridge/2.0" schemaLocation="http://schemas.opengis.net/citygml/bridge/2.0/bridge.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/building/2.0" schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityfurniture/2.0" schemaLocation="http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityobjectgroup/2.0" schemaLocation="http://schemas.opengis.net/citygml/cityobjectgroup/2.0/cityObjectGroup.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/generics/2.0" schemaLocation="http://schemas.opengis.net/citygml/generics/2.0/generics.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/landuse/2.0" schemaLocation="http://schemas.opengis.net/citygml/landuse/2.0/landUse.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/relief/2.0" schemaLocation="http://schemas.opengis.net/citygml/relief/2.0/relief.xsd"/>
</xs:schema>
```
The following excerpt of a CityGML dataset exemplifies how to apply the base profile schema CityGML.xsd to a CityGML instance document. The dataset contains two building objects and a city object group. The base profile defined by CityGML.xsd is referenced using the xsi:schemaLocation attribute of the root element. Thus, all CityGML modules are employed by the instance document and no further references to the XML Schema documents of the CityGML modules are necessary.
8 Spatial 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. 9 and Fig. 10. 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 8.2).

8.1 Geometric-topological model

The geometry model of GML3 consists of primitives, which may be combined to form complexes, composite geometries or aggregates. For each dimension, there is a geometrical object: a zero-dimensional object is a Point, a one-dimensional a Curve, a two-dimensional a Surface, and a three-dimensional a Solid (Fig. 9). Each geometry can have its own coordinate reference system. A solid is bounded by surfaces and a surface by curves. In CityGML, 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.

![UML diagram of CityGML’s geometry model (subset and profile of GML3): Primitives and Composites.](image)

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in Fig. 11). For 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, and a MultiSolid (see Fig. 10). In contrast to aggregates, a complex is topologically structured: its parts must be disjoint, 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. (cf. Fig. 9).
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 contour characteristics of the terrain.

The GML3 composite model realises 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 modelling of topology, for example 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. 12, 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 annex G.4.

The following excerpt of a CityGML example file defines a gml:Polygon with a gml:id wallSurface4711, which is part of the geometry property lod2Solid of a building. Another building being adjacent to the first building references this shared polygon in its geometry representation.

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

Fig. 12: Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).
8.2  Spatial reference system

When dealing with geoinformation and virtual 3D city models in particular, the exact spatial reference is of utmost importance and a key requirement for the integration of different spatial datasets in a single 3D city model. CityGML inherits GML3’s spatial capabilities of handling Coordinate Reference Systems (CRS) which is the usual way of denoting the spatial reference in GML 3.1.1. As CityGML is a true 3D standard, geometry elements are associated with a 3D CRS. There are only few exceptions to this rule where CityGML allows a 2D geometry element (for example, the referencePoint of a GeoreferencedTexture defined in CityGML’s Appearance module must be given with 2D coordinate values, cf. chapter 9.4).

In general, a geometry may point to the CRS definition used by this geometry through the attribute srsName which is inherited from the abstract GML superclass gml:Geometry. This may be a reference to a well-known CRS definition provided by an authority organization such as the European Petroleum Survey Group (EPSG), but may also be a pointer to a CRS that is locally defined within the same CityGML instance document. The OGC document “Definition identifier URNs in OGC namespace” (cf. Whiteside 2009; OGC Doc. No. 07-092r3) provides best practices for the URN encoding of CRS references. Amongst others, it describes how to reference a single well-known 3D CRS definition (such as a 3D geographic CRS) as well as a compound CRS which combines two or more well-known CRS definitions (e.g., a projected CRS for the planimetry with a vertical CRS for the height reference). Examples for denoting a compound CRS for a CityGML instance document are given in Annex G.

GML3 also supports the definition of engineering CRSs which are used in a contextually local sense. For example, this might be a local 3D Cartesian coordinate system that is essentially based on a flat-earth approximation of the earth’s surface, and thus ignores the effect of earth curvature on feature geometry (cf. chapter 12.1.4.4 of the GML 3.1.1 specification document). Local engineering CRSs are commonly applied in the AEC/FM domain and thus are useful when integrating CAD data or BIM models into a 3D city model. Annex G.9 provides an example demonstrating the definition of an engineering CRS within a CityGML instance document and the use of local coordinate values for the feature geometry. The definition of an engineering CRS requires an anchor point which relates the origin of the local coordinate system to a point on the earth’s surface in order to facilitate the transformation of coordinates from the local engineering CRS.

According to GML 3.1.1, if no srsName attribute is given on a geometry element, then the CRS shall be specified as part of the larger context this geometry element is part of, e.g. a geometric aggregate. For convenience in constructing feature and feature collection instances, the value of the srsName attribute on the gml:Envelope (or gml:Box) which is the value of the gml:boundedBy property of the feature shall be inherited by all directly expressed geometries in all properties of the feature or members of the collection, unless overruled by the presence of a local srsName. Thus it is not necessary for a geometry to carry an srsName attribute if it uses the same CRS as given on the gml:boundedBy property of its parent feature. Inheritance of the CRS continues to any depth of nesting, but if overruled by a local srsName declaration, then the new CRS is inherited by all its children in turn (cf. chapter 8.3 of the GML 3.1.1 specification document).

It is strongly recommended that any CityGML instance document explicitly specifies the CRS for all contained geometry elements. This is especially important if the instance document is to be exchanged externally with third parties or is to be integrated with other spatial datasets. A mixed usage of different CRSs within the same dataset is possible and conformant with GML 3.1.1, whereas a single CRS reference given on the embracing CityModel feature collection (cf. chapter 10.1) simplifies the processing of the dataset by software systems. As for CityGML 2.0, this recommendation is non-normative and thus not accompanied by a conformance class. The main reason for this is to maintain backwards compatibility with CityGML 1.0.

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, for example, used in CityGML’s building, bridge, tunnel, and vegetation model as well as for city furniture and generic objects. Implicit geometries may be applied to features from different thematic fields of CityGML in order to geometrically represent the features within a specific level of detail (LOD). Thus, each extension module may define spatial properties providing implicit geometries for its thematic classes. For this reason, the concept of implicit geometries is defined within the CityGML core module (cf. chapter 10.1). However, its description is drawn here
since implicit geometries are part of CityGML’s spatial model. The UML diagram is depicted in Fig. 13. The corresponding XML schema definition is provided in annex A.1.

An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, for example a tree or other vegetation objects, a traffic light or a traffic sign. This prototypic geometry object is reused 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 modelled by the transformation matrix.

Fig. 13: UML diagram of ImplicitGeometries. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Core module.

The reason for using the concept of implicit geometries in CityGML is space efficiency. Since the shape of, for example, 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/x3d+xml” for X3D models).

The implicit representation of 3D object geometry has some advantages compared to the explicit modelling, 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 visualisation 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. 65).

**XML namespace**

The XML namespace of the CityGML Core module defining the concept of implicit geometries is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/2.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

**ImplicitGeometryType, ImplicitRepresentationPropertyType**

```
<xs:complexType name="ImplicitGeometryType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType">
      <xs:sequence>
        <xs:element name="mimeType" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="transformationMatrix" type="TransformationMatrix4x4Type" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
8.3.1 Code lists

The `mimeType` attribute of `ImplicitGeometry` is specified as `gml:CodeType`. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

8.3.2 Example CityGML datasets

An example for an implicit geometry is given by the following city furniture object (cf. chapter 10.9), which is represented by a geometry in LOD2:

The shape of the geometry of the traffic light (city furniture with class “1000” and function “1080” according to the code lists proposed in annex C.4) 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:
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 \((\cos(30) = 0.866025\) 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 \((5793898.77, 3603845.54, 44.8, 1)^T\) is added, yielding the actual geometry of the city furniture.

8.3.3 Conformance requirements

Base requirements

1. In order to geometrically represent a feature using the concept of implicit geometries, the corresponding thematic class of the feature shall define a spatial property of the type `ImplicitRepresentationPropertyType`. Thus, for all CityGML extension modules only the type `ImplicitRepresentationPropertyType` shall be used for spatial properties providing implicit geometries.

2. If the shape of an implicit geometry is referenced by an URI using the `libraryObject` property (type: `xs:anyURI`) of the element `ImplicitGeometry`, also the MIME type of the denoted object must be specified.

Referential integrity

3. The type `ImplicitRepresentationPropertyType` may contain an `ImplicitGeometry` element inline or an XLink reference to a remote `ImplicitGeometry` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the corresponding property of type `ImplicitRepresentationPropertyType` may only point to a remote `ImplicitGeometry` element (where remote `ImplicitGeometry` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
9 Appearance model

In addition to spatial properties, CityGML features have appearances – observable properties of the feature’s surface. Appearances are not limited to visual data but represent arbitrary categories called themes such as infrared radiation, noise pollution, or earthquake-induced structural stress. Each LOD can have an individual appearance for a specific theme. An appearance is composed of data for each surface geometry object, i.e. surface data. A single surface geometry object may have surface data for multiple themes. Similarly, surface data can be shared by multiple surface geometry objects (e.g. road paving). Finally, surface data values can either be constant across a surface or depend on the exact location within the surface.

CityGML’s appearance model is defined within the extension module Appearance (cf. chapter 7). The UML diagram of the appearance model is illustrated in Fig. 14, for the XML Schema definition see annex A.2.

Fig. 14: UML diagram of CityGML’s appearance model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Appearance module.

In CityGML’s appearance model, themes are represented by an identifier only. The appearance of a city model for a given theme is defined by a set of Appearance objects referencing this theme. Thus, the Appearance objects
belonging to the same theme compose a virtual group. They may be included in different places within a CityGML dataset. Furthermore a single CityGML dataset may contain several themes. An Appearance object collects surface data relevant for a specific theme either for individual features or the whole city model in any LOD. Surface data is represented by objects of class _SurfaceData and its descendents with each covering the whole area of a surface geometry object. The relation between surface data and surface geometry objects is expressed by an URI (Uniform Resource Identifier) link from a _SurfaceData object to an object of type gml:AbstractSurfaceType or type gml:MultiSurface.

A constant surface property is modelled as material. A surface property, which depends on the location within the surface, is modelled as texture. Each surface geometry object can have both a material and a texture per theme and side. This allows for providing both a constant approximation and a complex measurement of a surface’s property simultaneously. An application is responsible for choosing the appropriate property representation for its task (e.g. analysis or rendering). A specific mixing is not defined since this is beyond the scope of CityGML. If a surface geometry object is to receive multiple textures or materials, each texture or material requires a separate theme. The mixing of themes or their usage is not defined within CityGML and left to the application.

XML namespace

The XML namespace of the CityGML Appearance module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/appearance/2.0. Within the XML Schema definition of the Appearance module, this URI is also used to identify the default namespace.

9.1 Relation between appearances, features and geometry

Despite the close relation between surface data and surface, surface data is stored separately in the feature to preserve the original GML geometry model. Instead of surface data being an attribute of the respective target surface geometry object, each surface data object maintains a set of URIs specifying the gml:ids of the target surface geometry objects (of type gml:AbstractSurfaceType or gml:MultiSurface). In case of a composite or aggregate target surface, the surface data object is assigned to all contained surfaces. Other target types such as features, solids, or gml:AbstractSurfacePatchType (which includes gml:Triangle) are invalid, even though the XML schema language cannot formally express constrains on URI target types. For the exact mapping function of surface data values to a surface patch refer to the respective surface data type description.

The limitation of valid target types to gml:AbstractSurfaceType and gml:MultiSurface excluding gml:AbstractSurfacePatchType is based on the GML geometry model and its use in CityGML. In general, GML surfaces are represented using subclasses of gml:AbstractSurfaceType. Such surfaces are required to be continuous. A gml:MultiSurface does not need to fulfill this requirement and consequently is no gml:AbstractSurfaceType (cf. 8.1). Since captured real-world surfaces often cannot be guaranteed to be continuous, CityGML allows for gml:MultiSurface to represent a feature’s boundary in various places as an alternative to a continuous surface. To treat such surfaces similarly to a gml:CompositeSurface, surface data objects are allowed to link to gml:MultiSurface objects. gml:AbstractSurfacePatchType is no valid target type since it is not derived from gml:AbstractGMLType. Thus, a gml:AbstractSurfacePatchType (which includes gml:Triangle and gml:Rectangle) cannot receive a gml:id and cannot be referenced.

Each surface geometry object can have per theme at most one active front-facing material, one active back-facing material, one active front-facing texture, and one active back-facing texture. If multiple surface data objects of the same category and theme are assigned to a surface geometry object, one is chosen to become active. Multiple indirect assignments due to nested surface definitions are resolved by overwriting, e.g. the front-facing material of a gml:Polygon becomes active by overwriting the front-facing material of the parental gml:CompositeSurface. Multiple direct assignments, i.e. a surface geometry object’s gml:id is referenced multiple times within a theme, are not allowed and are resolved implementation-dependently by choosing exactly one of the conflicting surface data objects. Thus, multiple direct assignments within a theme need to be avoided.

Each _CityObject feature can store surface data. Thus, surface data is arranged in the feature hierarchy of a CityGML dataset. Surface data then links to its target surface using URIs. Even though the linking mechanism permits arbitrary links across the feature hierarchy to another feature’s surface, it is recommended to follow the principle of locality: Surface data should be stored such that the linked surfaces only belong to the containing _CityObject feature and its children. “Global” surface data should be stored with the city model. Adhering to the
locality principle also ensures that CityObjects retrieved from a WFS will contain the respective appearance information.

The locality principle allows for the following algorithm to find all relevant _SurfaceData objects referring to a given surface geometry object (of type gml:AbstractSurfaceType or gml:MultiSurface) in a given _CityObject:

```plaintext
function findSurfaceData
  in:  gmlSurface, cityObject
  out: frontMaterial, frontTexture, backMaterial, backTexture
1:   frontMaterial := empty
2:   frontTexture := empty
3:   backMaterial := empty
4:   backTexture := empty
5:   flip := false
6:   while (gmlSurface) {  // traverse the geometry hierarchy from inner to outer
7:     cObj := cityObject  // start from the innermost cityobject
8:     while (cObj) {      // traverse the cityobject hierarchy for the current geometry object
9:       // search all surfaceData objects in all appearance containers
10:      foreach (appearance in cObj) {
11:        foreach (surfaceData in appearance) {
12:          if (surfaceData refers to gmlSurface) {  // if a surfaceData object refers to the geometry object, check its category
13:            if (flip) {  // consider flipping
14:              // only pick the first surfaceData for a particular category
15:                if (surfaceData is frontside material AND backMaterial is empty) {
16:                  backMaterial := surfaceData
17:                }
18:                if (surfaceData is frontside texture AND backTexture is empty) {
19:                  backTexture := surfaceData
20:                }
21:                if (surfaceData is backside material AND frontMaterial is empty) {
22:                  frontMaterial := surfaceData
23:                }
24:                if (surfaceData is backside texture AND frontTexture is empty) {
25:                  frontTexture := surfaceData
26:                }
27:                }
28:              } else {
29:                // only pick the first surfaceData for a particular category
30:                if (surfaceData is frontside material AND frontMaterial is empty) {
31:                  frontMaterial := surfaceData
32:                }
33:                if (surfaceData is frontside texture AND frontTexture is empty) {
34:                  frontTexture := surfaceData
35:                }
36:                if (surfaceData is backside material AND backMaterial is empty) {
37:                  backMaterial := surfaceData
38:                }
39:                if (surfaceData is backside texture AND backTexture is empty) {
40:                  backTexture := surfaceData
41:                }
42:              }
43:            } else {
44:              // shortcut: could stop here if all 4 categories have been found
45:            }
46:          }
47:        }
48:        cObj := cObj.parent  // this also includes the global CityModel
49:     }
50:   }  // this also includes a root gml:MultiSurface
51:  gmlSurface := gmlSurface.parent  // this also includes a root gml:MultiSurface
52:  if (gmlSurface isA gml::OrientableSurface AND gmlSurface.orientation is negative) {
53:    negate flip
54:  }
55: }
```

Listing 1: Algorithm to find all relevant _SurfaceData objects referring to a given surface geometry object (of type gml:AbstractSurfaceType or gml:MultiSurface) in a given _CityObject.
The evaluation of the isFront property of a _SurfaceData object needs to take gml:OrientableSurfaces into account, as those can flip the orientation of a surface. Assume a gml:OrientableSurface os, which flips its base surface bs. A front side texture t targeting bs will appear on the actual front side of bs. If t targets os, it will appear on the back side of bs. If t targets both os and bs, it appears on both sides of bs since it becomes the front and back side texture.

XLinks influence the hierarchy traversal in the pseudocode. In general, the separation of surface data and geometry objects requires the reevaluation of the surface data assignment for each occurrence of a geometry object in the context of the respective _CityObject. Stepping up the (geometry or _CityObject) hierarchy in the algorithm takes XLinks into account, i.e., for the purpose of this algorithm, referenced objects are conceptually copied to the location of the referring XLink. In particular, this applies to ImplicitGeometry objects. If an ImplicitGeometry object contains GML geometry (in the relativeGMLGeometry property), the surface data assignment needs to be reevaluated in the context of each referring _CityObject. Thus, the appearance (but not the relative geometry) of a given ImplicitGeometry can differ between its occurrences. A consistent appearance results if all required surface data objects are placed in Appearance objects and the latter are stored either

1. in the _CityObject containing the original ImplicitGeometry with XLinks referencing the same Appearance objects in all _CityObjects that refer to the ImplicitGeometry or
2. in the global CityModel.

### 9.2 Appearance and SurfaceData

The feature class Appearance defines a container for surface data objects. It provides the theme that all contained surface data objects are related to. All appearance objects with the same theme in a CityGML file are considered a group. Surface data objects are stored in the surfaceDataMember property. They can be used in multiple themes simultaneously as remote properties.

The feature class _SurfaceData is the base class for materials and textures. Its only element is the boolean flag isFront, which determines the side a surface data object applies to. Please note, that all classes of the appearance model support CityGML’s ADE mechanism (cf. chapters 6.12 and 10.13). The hooks for application specific extensions are realized by the elements “_GenericApplicationPropertyOf…”.

**AppearanceType, Appearance, AppearancePropertyType**

```xml
<xs:complexType name="AppearanceType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="theme" type="xs:string" minOccurs="0"/>
        <xs:element name="surfaceDataMember" type="SurfaceDataPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

**appearanceMember, appearance**

```xml
<xs:element name="appearanceMember" type="gml:FeaturePropertyType Group="gml:featureMember"/>
<!--------------------------------------------------------------------------->
<xs:element name="appearance" type="AppearancePropertyType Group="core:_GenericApplicationPropertyOfCityObject"/>
<!--------------------------------------------------------------------------->
```
The definition of appearanceMember allows for an arbitrary or even mixed sequence of _CityObject features and Appearance features within a CityModel feature collection (cf. chapter 10.1).

In order to store appearance information within a single _CityObject feature, the corresponding abstract class _CityObject of the core module is augmented by the property element appearance. The additional property appearance is injected into _CityObject using CityGML’s Application Domain Extension mechanism (cf. chapter 10.13). By this means, each thematic subclass of _CityObject inherits this property. Thus, the Appearance module has a deliberate impact on each extension module defining thematic subclasses of _CityObject.

### 9.3 Material

Materials define light reflection properties being constant for a whole surface geometry object. The definition of the class X3DMaterial is adopted from the X3D and COLLADA specification (cf. X3D, COLLADA specification). diffuseColor defines the color of diffusely reflected light. specularColor defines the color of a directed reflection. emissiveColor is the color of light generated by the surface. All colors use RGB values with red, green, and blue between 0 and 1. Transparency is defined separately using the transparency element where 0 stands for fully opaque and 1 for fully transparent. ambientIntensity defines the minimum percentage of diffuseColor that is visible regardless of light sources. shininess controls the sharpness of the specular highlight. 0 produces a soft glow while 1 results in a sharp highlight. isSmooth gives a hint for normal interpolation. If this boolean flag is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading).

Target surfaces are specified using target elements. Each element contains the URI of one target surface geometry object (of type gml:AbstractSurfaceType or gml:MultiSurface).

### X3DMaterialType, X3DMaterial

```xml
<xs:complexType name="X3DMaterialType" base="gml:AbstractFeatureType">
  <xs:sequence>
    <xs:element name="ambientIntensity" type="core:doubleBetween0and1" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="diffuseColor" type="Color" default="0.8 0.8 0.8" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="emissiveColor" type="Color" default="0.0 0.0 0.0" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="specularColor" type="Color" default="1.0 1.0 1.0" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="transparency" type="core:doubleBetween0and1" default="0.2" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="isSmooth" type="xs:boolean" default="false" minOccurs="0" maxOccurs="unbounded" />
    <xs:element ref="_GenericApplicationPropertyOfX3DMaterial" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```
9.4 Texture and texture mapping

The abstract base class for textures is _Texture. Textures in CityGML are always raster-based 2D textures. The raster image is specified by imageURI using a URI and can be an arbitrary image data resource, even a preformatted request for a web service. The image data format can be defined using standard MIME types in the mimeType element.

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.

The specification of texture wrapping is adopted from the COLLADA standard. Texture wrapping is required when accessing a texture outside the underlying image raster. wrapMode can have one of five values (Fig. 15 illustrates the effect of these wrap modes):

1. none — the resulting color is fully transparent
2. wrap — the texture is repeated
3. mirror — the texture is repeated and mirrored
4. clamp — the texture is clamped to its edges
5. border — the resulting color is specified by the borderColor element (RGBA)

In wrap mode mirror, the texture image is repeated both in horizontal and in vertical direction to fill the texture space similar to wrap mode wrap. Unlike wrap, each repetition results from flipping the previous texture part along the repetition direction. This behaviour removes the edge correspondence constraint for wrapped textures and always results in a seamless texture.

![Fig. 15: A texture (a) applied to a facade using different wrap modes: (b) none, (c) wrap, (d) mirror, (e) clamp and (f) border. The border color is red. The numbers denote texture coordinates (image: Hasso-Plattner-Institute).](image)

AbstractTextureType, _Texture, WrapModeType, TextureTypeType
_Texture_ is further specialised according to the texture parameterisation, i.e., the mapping function from a location on the surface to a location in the texture image. CityGML uses the notion of texture space, where the texture image always occupies the region \([0,1]^2\) regardless of the actual image size or aspect ratio. The lower left image corner is located at the origin (some graphics APIs may use other conventions and require texture coordinate conversion). The mapping function must be known for each surface geometry object to receive texture.

The class _GeoreferencedTexture_ describes a texture that uses a planimetric projection. Consequently, it does not make sense to texture vertical surfaces using a _GeoreferencedTexture_. Such a texture has a unique mapping function which is usually provided with the image file (e.g., georeferenced TIFF) or as a separate ESRI world file\(^1\). The search order for an external georeference is determined by the boolean flag _preferWorldFile_. If this flag is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the image data is used. If _preferWorldFile_ is false, the world file is used only if no georeference from the image data is available.

Alternatively, CityGML allows for inline specification of a georeference similar to a world file. This internal georeference specification always takes precedence over any external georeference. _referencePoint_ defines the location of the center of the upper left image pixel in world space and corresponds to values 5 and 6 in an ESRI world file. Since _GeoreferencedTexture_ uses a planimetric projection, _referencePoint_ is two-dimensional. _orientation_ defines the rotation and scaling of the image in form of a 2x2 matrix (a list of 4 doubles in row-major order corresponding to values 1, 3, 2, and 4 in an ESRI world file). The CRS of this transformation is identical to

---

\(^1\) Further information about the ESRI world file format is provided at [http://en.wikipedia.org/wiki/World_file](http://en.wikipedia.org/wiki/World_file).
the referencePoint’s CRS. A planimetric point \((x, y)\) in that CRS is transformed to a point \((s, t)\) in texture space using the formula:

\[
\begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} 1/w & 0 \\ 0 & -1/h \end{bmatrix} \cdot M^{-1} \cdot \begin{bmatrix} x \\ y \end{bmatrix} - P_R + \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

with \(M\) denoting orientation, \(P_R\) denoting referencePoint., \(w\) the image’s width in pixels, and \(h\) the image’s height in pixels. This transformation compensates for the difference between the image coordinate system used in ESRI world files (origin in upper left corner, positive x-axis rightwards, and positive y-axis downwards) and texture space in CityGML (origin in lower left corner, positive x-axis rightwards, and positive y-axis upwards).

If neither an internal nor an external georeference is given the GeoreferencedTexture is invalid. Each target surface geometry object is specified by an URI in a target element. All target surface geometry objects share the mapping function defined by the georeference. No other mapping function is allowed. Please note, that the gml:boundedBy property inherited from gml:AbstractFeatureType could be set to the bounding box of valid image data to allow for spatial queries. Fig. 16 shows a georeferenced texture applied to the ground and all roof surfaces.

The class ParameterizedTexture describes a texture with target-dependent mapping function. The mapping is defined by subclasses of class _TextureParameterization as a property of the link to the target surface geometry object. Each target surface geometry object is specified as URI in the uri attribute of a separate target element. Since target implements gml:AssociationAttributeGroup, it allows referencing to a remote _TextureParameterization object (using the xlink:href attribute), e.g. for sharing a mapping function between targets or textures in different themes. The mapping function can either use the concept of texture coordinates (through class TexCoordList) or a transformation matrix from world space to texture space (through class TexCoordGen).

![Fig. 17: Positioning of textures using texture coordinates (image: IGG Uni Bonn).](image-url)
Texture coordinates are applicable only to polygonal surfaces, whose boundaries are described by gml:LinearRing (e.g., gml:Triangle, gml:Polygon, or a gml:MultiSurface consisting of gml:Polygons). They define an explicit mapping of a surface’s vertices to points in texture space, i.e. each vertex including interior ring vertices must receive a corresponding coordinate pair in texture space (for the notion of coordinates, refer to ISO 19111). These coordinates are not restricted to the [0,1] interval. Texture coordinates for interior surface points are planarly interpolated from the vertices’ texture coordinates. Fig. 16 shows an example.

Texture coordinates for a target surface geometry object are specified using class TexCoordList as a texture parameterization object in the texture’s target property. Each exterior and interior gml:LinearRing composing the boundary of the target surface geometry object (which also might be a gml:CompositeSurface, gml:MultiSurface, or gml:TriangulatedSurface) requires its own set of texture coordinates. A set of texture coordinates is specified using the textureCoordinates element of class TexCoordList. Thus, a TexCoordList contains as many textureCoordinate elements as the target surface geometry object contains gml:LinearRings.

textureCoordinate’s mandatory attribute ring provides the gml:id of the respective ring. The content is an ordered list of double values where each two values define a \((s,t)\) texture coordinate pair with \(s\) denoting the horizontal and \(t\) the vertical texture axis. The list contains one pair per ring point with the pairs’ order corresponding to the ring points’ order in the CityGML document (regardless of a possibly flipped surface orientation). If any ring point of a target surface geometry object has no texture coordinates assigned, the mapping is incomplete and the respective surface cannot be textured. In case of aggregated target geometry objects, mapping completeness is determined only for leaf geometry objects.

Alternatively, the mapping function can comprise a 3x4 transformation matrix specified by class TexCoordGen. The transformation matrix, specified by the worldToTexture element, defines a linear transformation from a spatial location in homogeneous coordinates to texture space. The use of homogeneous coordinates facilitates perspective projections as transformation, e.g. for projecting a photograph into a city model (cf. Fig. 18). Texture coordinates \((s,t)\) are calculated from a space location \((x,y,z)\) as \((s,t) = (s'/q', t'/q')\) with \((s',t',q') = M \cdot (x,y,z,1)^T\). \(M\) denotes the 3x4 transformation matrix. Compared to a general 4x4 transformation, the resulting \(z\) component is ignored. Thus, the respective matrix row is omitted. Additionally, the worldToTexture element uses the gml:SRSReferenceGroup attributes to define its CRS. A location in world space has to be first transformed into this CRS before the transformation matrix can be applied.

The following construction results in a worldToTexture transformation that mimics the process of taking a photograph by projecting a location in world space (in the city model) to a location in texture space:

\[
M = \begin{bmatrix} 0.5 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 1 & 0 \\ \end{bmatrix} \cdot \begin{bmatrix} 2f/w & 0 & 0 & 0 \\ 0 & 2f/h & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \end{bmatrix} \cdot \begin{bmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ d_x & d_y & d_z & 0 \\ 0 & 0 & 0 & 1 \\ \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \\ \end{bmatrix}
\]
In this formula, \( f \) denotes the focal length; \( w \) and \( h \) represent the image sensor’s physical dimensions; \( \vec{r} \), \( \vec{u} \), and \( \vec{d} \) define the camera’s frame of reference as right, up and directional unit vectors expressed in world coordinates; and \( P \) stands for the camera’s location in world space. Fig. 19 sketches this setting.

![Diagram](image.png)

Fig. 19: Projective texture mapping. All points on a ray \( R \) starting from the projection center \( P \) are mapped to the same point \( T \) in texture space (image: Hasso-Plattner-Institute, IGG TU Berlin).

Alternatively, if the 3x4 camera matrix \( M_F \) is known (e.g. through a calibration and registration process), it can easily be adopted for use in `worldToTexture`. \( M_F \) is derived from intrinsic and extrinsic camera parameters (interior and exterior orientation) and transforms a location in world space to a pixel location in the image. Assuming the upper left image corner has pixel coordinates \((0,0)\), the complete transformation to texture space coordinates can be written as \((\text{width}_{\text{image}}\text{ and } \text{height}_{\text{image}} \text{ denote the image size in pixels})\):

\[
M = \begin{pmatrix}
\frac{1}{\text{width}_{\text{image}}} & 0 & 0 & 0 \\
0 & -\frac{1}{\text{height}_{\text{image}}} & 1 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix} \cdot M_F
\]

Please note, that `worldToTexture` cannot compensate for radial or other non-linear distortions introduced by a real camera lens.

Another use of `worldToTexture` is texturing a facade with complex geometry without specifying texture coordinates for each \texttt{gml:LinearRing}. Instead, only the facade’s aggregated surface becomes the texture target using a `TexCoordGen` as parameterization. Then, `worldToTexture` effectively encodes an orthographic projection of world space into texture space. For the special case of a vertical facade this transformation is given by:

\[
M = \begin{pmatrix}
\frac{1}{\text{width}_x} & 0 & 0 & 0 \\
0 & \frac{1}{\text{height}_y} & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
-n_y & n_x & 0 & 0 \\
0 & 0 & 1 & 0 \\
n_y & n_x & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & -F_z \\
0 & 1 & 0 & -F_y \\
0 & 0 & 1 & -F_x \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

This equation assumes \( \vec{n} \) denoting the facade’s overall normal vector (normalized, pointing outward, and being parallel to the ground), \( F \) denoting the facade’s lower left point, and \( \text{width}_x \) and \( \text{height}_y \) specifying the facade’s dimensions in world units. For the general case of an arbitrary normal vector the facade orientation matrix assumes a form similar to the camera orientation matrix:

\[
M = \begin{pmatrix}
\frac{1}{\text{width}_x} & 0 & 0 & 0 \\
0 & \frac{1}{\text{height}_y} & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
r_x & r_y & r_z & 0 \\
u_x & u_y & u_z & 0 \\
n_x & n_y & n_z & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & -F_z \\
0 & 1 & 0 & -F_y \\
0 & 0 & 1 & -F_x \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[\vec{r} = \left(\begin{array}{c}0 \\ 0 \\ 1\end{array}\right) \times \vec{n}\]

\[\vec{u} = \vec{n} \times \vec{r}\]

\[\vec{P}_{\text{image}} = \left(\begin{array}{c}0 \\ 0 \\ \text{image size in pixels}\end{array}\right)\]
ParameterizedTextureType, ParameterizedTexture, TextureAssociationType

<xs:complexType name="ParameterizedTextureType">
  <xs:complexContent>
    <xs:extension base="AbstractTextureType">
      <xs:sequence>
        <xs:element name="target" type="TextureAssociationType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

AbstractTextureParameterizationType, TexCoordListType, TexCoordGenType

<xs:complexType name="AbstractTextureParameterizationType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType">
      <xs:sequence>
        <xs:element name="textureCoordinates" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="TexCoordList" type="TexCoordListType" substitutionGroup="_TextureParameterization"/>

<xs:element name="TexCoordGen" type="TexCoordGenType" substitutionGroup="gml:_GML"/>

<xs:element name="worldToTexture" type="core:TransformationMatrix3x4Type"/>
9.5 Related concepts

The notion of appearance clearly relates to the generic coverage approach (cf. ISO 19123 and OGC Abstract specification, Topic 6). Surface data can be described as discrete or continuous coverage over a surface as two-dimensional domain with a specific mapping function. Such an implementation requires the extension of GML coverages (as of version 3.1) by suitable mapping functions and specialisation for valid domain and range sets. For reasons of simplicity and comprehensibility both in implementation and usage, CityGML does not follow this approach, but relies on textures and materials as well-known surface property descriptions from the field of computer graphics (cf. X3D, COLLADA specification, Foley et al.). Textures and materials store data as color using an appropriate mapping. If such a mapping is impractical, data storage can be customised using ADEs. A review of coverages for appearance modelling is considered for CityGML beyond version 2.0.0.

Appearance is also related to portrayal. Portrayal describes the composition and symbolisation of a digital model’s image, i.e. presentation, while appearance encodes observations of the real object’s surface, i.e. data. Even though being based on graphical terms such as textures and materials, surface data is not limited to being input for portrayal, but similarly serves as input or output for analyses on a feature’s surface. Consequently, CityGML does not define mixing or composition of themes for portrayal purposes. Portrayal is left to viewer applications or styling specification languages such as OGC Styled Layer Descriptors (SLD) or OGC Symbol Encoding (SE).

9.6 Code lists

The mimeType attribute of the feature _Texture is specified as gml:CodeType. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

9.7 Conformance requirements

Base requirements

1. A surface geometry object may be the target of at most two textures and two materials (one for front and back respectively) per theme.

2. The referencePoint property (type: gml:PointPropertyType) of the element GeoreferencedTexture may only contain or reference a point geometry object with 2D coordinate values.

3. Texture coordinates given by the textureCoordinates property of the element TexCoordList define an explicit mapping of a surface’s boundary points to points in texture space. Each boundary point of the surface must receive a corresponding coordinate pair in texture space. The coordinate pair in texture space shall be given as two doubles per boundary point. The order of the coordinate pairs must follow the order of the boundary points in the CityGML document (regardless of a possibly flipped surface orientation). Each gml:LinearRing composing the boundary of the target surface geometry object requires its own set of texture coordinates.

4. A GeoreferencedTexture element must provide either internal or external georeference, otherwise it is invalid. Internal georeference shall be declared by the referencePoint property (type: gml:PointPropertyType) and the orientation property (type: core:TransformationMatrix2x2Type) of the
element GeoreferencedTexture. External georeference may be provided by the texture image file itself (e.g. GeoTIFF) or by an accompanying world file.

Referential integrity

5. The appearanceMember element (type: AppearancePropertyType) may contain an Appearance element inline or an XLink reference to a remote Appearance element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the appearanceMember element may only point to a remote Appearance element (where remote Appearance elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

6. The appearance property (type: AppearancePropertyType) of the element core:_CityObject may contain an Appearance element inline or an XLink reference to a remote Appearance element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the appearance property may only point to a remote Appearance element (where remote Appearance elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

7. The surfaceDataMember property (type: SurfaceDataPropertyType) of the element Appearance may contain a _SurfaceData element inline or an XLink reference to a remote _SurfaceData element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the surfaceDataMember property may only point to a remote _SurfaceData element (where remote _SurfaceData elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

8. The target property (type: TextureAssociationType) of the element ParameterizedTexture may contain a _TextureParameterization element inline or an XLink reference to a remote _TextureParameterization element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the target property may only point to a remote _TextureParameterization element (where remote _TextureParameterization elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

9. The target property (type xs:anyURI) of the element GeoreferencedTexture shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.

10. The uri attribute of the complex type TextureAssociationType shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.

11. The ring attribute of the textureCoordinates property of the element TexCoordList shall specify the gml:id of the target surface geometry object which may only be of type gml:LinearRing.

12. The target property (type xs:anyURI) of the element X3DMaterial shall specify the gml:id of the target surface geometry object which may only be of type gml:AbstractSurfaceType or gml:MultiSurface.
9.8 Material model of previous CityGML versions [deprecated]

Since GML3 has no built-in concept for the representation of surface materials, previous versions of CityGML extend the GML3 geometry model by the class TexturedSurface, which allows for assigning appearance properties (colors, shininess, transparency) and textures to 3D surfaces. The definition of the appearance properties is adopted from the X3D specification. This approach for appearance modelling has been deprecated due to inherent limitations. However, in order to provide a certain degree of backwards compatibility for already existing CityGML implementations, the approach has been incorporated into CityGML version 1.0 and version 2.0 as a separate extension module called TexturedSurface. By this means, implementations may employ the old material model by supporting this module. Please note, that appearance information modelled according to the TexturedSurface module can be converted without information loss to the concepts provided by CityGML's Appearance module that has been introduced in the previous clauses of this chapter. Thus, the use of the TexturedSurface module is strongly discouraged and implementations should only stick to the Appearance module instead. Moreover, the TexturedSurface module is expected to be removed in future versions of CityGML.

For the TexturedSurface module, each surface or composite surface can be specialized to a TexturedSurface, which can be assigned Materials (colors, shininess, transparency) or SimpleTextures. Fig. 20 depicts the UML diagram, for XML schema definition see annex A.14.

Fig. 20: UML diagram of CityGML's material model. Please note, that this approach for appearance modelling has been marked as deprecated and is expected to be removed in future CityGML versions. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML TexturedSurface module.

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

A texture is specified as a raster image referenced by an URI (Uniform Resource Identifier) and can be an arbitrary resource, even on 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. 17). 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 also to be in the range of 0 to 1, 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 for 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 parent class of `Material` and `SimpleTexture`.

**XML namespace**

The XML namespace of the CityGML `TexturedSurface` module is identified by the Uniform Resource Identifier (URI) `http://www.opengis.net/citygml/texturedsurface/2.0`. Within the XML Schema definition of the `TexturedSurface` module, this URI is also used to identify the default namespace.

### 9.8.1 Textured surfaces

**TexturedSurfaceType, TexturedSurface, AppearancePropertyType**

```xml
<xs:complexType name="TexturedSurfaceType">
  <xs:complexContent>
    <xs:extension base="gml:OrienteableSurfaceType">
      <xs:sequence>
        <xs:element ref="appearance" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

TexturedSurface may have one or more appearance properties, which can either be a `Material` (Color,...) or a `SimpleTexture`. The `appearance` element can either be represented inline 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 (type `gml:SignType`) of the appearance property element, which refers to the corresponding `orientation` attribute of the orientable surface: + means the side with positive orientation and - the side with negative orientation.

**AbstractAppearanceType, Appearance**

```xml
<xs:complexType name="AbstractAppearanceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="gml:AbstractGMLType"/>
  </xs:complexContent>
</xs:complexType>

MaterialType, Material

```xml
<xs:complexType name="MaterialType">
  <xs:extension base="AbstractAppearanceType">
    <xs:sequence>
      <xs:element name="shininess" type="core:doubleBetween0and1" minOccurs="0"/>
      <xs:element name="transparency" type="core:doubleBetween0and1" minOccurs="0"/>
    </xs:sequence>
  </xs:extension>
</xs:complexType>
```
9.8.2 Conformance requirements

Referential integrity

- The appearance property (type: AppearancePropertyType) of the element TexturedSurface may contain an Appearance element inline or an XLink reference to a remote Appearance element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the appearance property may only point to a remote Appearance element (where remote Appearance elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10 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 important in many different application areas. 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, generalisations and associations.

The aim of the explicit modelling 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 standardised meaning or interpretation. In order to also allow for the exchange of objects and/or attributes that are not explicitly modelled in CityGML, the concepts of generic city objects and attributes as well as CityGML’s Application Domain Extension mechanism have been introduced (cf. chapter 10.12 and chapter 10.13).

Each field of CityGML’s thematic model is covered by a separate CityGML extension module. Thus, the extension modules are derived by vertically slicing the overall thematic data model of CityGML. All extension modules are based on and are dependent from the CityGML core module. The core comprises the basic concepts and components of the CityGML data model. Implementations may choose to combine CityGML extension modules in conjunction with the core according to their specific information needs or application domain. As for version 2.0 of CityGML, the following thirteen thematic extension modules are defined: Appearance, Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody, and TexturedSurface [deprecated]. Valid combinations of CityGML modules are called CityGML profiles. By this means, CityGML profiles explicitly allow for partial implementations of the overall CityGML data model (cf. chapter 7).

The thematic fields covered by the CityGML data model are introduced within the sub clauses of this chapter. Each sub clause is related to a specific CityGML module.
10.1 CityGML Core

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

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

Fig. 21: UML diagram of CityGML’s core module. The bracketed numbers following the attribute names denote the attribute’s 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. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Core module.

Fig. 22: Topological relations of a CityGML object with respect to a) the terrain and b) the water surface.

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

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

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

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

Fig. 23: CityGML’s top level class hierarchy. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the \texttt{CityGML Core} module.

Copyright © 2012 Open Geospatial Consortium.
The classes `GenericCityObject` and `_genericAttribute` defined within CityGML’s `Generics` module (cf. chapters 6.11 and 10.12) allow for modelling and exchanging of 3D objects which are not covered by any other thematic class or which require attributes not represented in CityGML. For example, in the future, sites derived from the abstract class `_Site` of the core module may be completed by further subclasses like excavation, city wall or embankment. At present, the class `GenericCityObject` should be used in order to represent and exchange these features. However, the concept of generic city objects and attributes may only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

If the `Generics` module is employed, each `CityObject` may be assigned an arbitrary number of generic attributes in order to represent additional properties of features. For this purpose, the `Generics` module augments the abstract base class `_CityObject` by the property element `_genericAttribute`. The additional property `_genericAttribute` is injected into `_CityObject` using CityGML’s Application Domain Extension mechanism (cf. chapter 10.13). By this means, each thematic subclass of `_CityObject` inherits this property and, thus, the possibility to contain generic attributes. Therefore, the `Generics` module has a deliberate impact on all CityGML extension modules defining thematic subclasses of `_CityObject`.

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

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

**XML namespace**

The XML namespace of the `CityGML Core` module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/2.0. Within the XML Schema definition of the core module, this URI is also used to identify the default namespace.

### 10.1.1 Base elements

**AbstractCityObjectType, _CityObject**

```xml
<xsd:complexType name="AbstractCityObjectType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element name="creationDate" type="xsd:date" minOccurs="0"/>
        <xsd:element name="terminationDate" type="xsd:date" minOccurs="0"/>
        <xsd:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="relativeToTerrain" type="RelativeToTerrainType" minOccurs="0"/>
        <xsd:element name="relativeToWater" type="RelativeToWaterType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```xml
<xsd:element name="_CityObject" type="AbstractCityObjectType" abstract="true" substitutionGroup="gml:Feature"/>
```

```xml
<xsd:element name="_GenericApplicationPropertyOfCityObject" type="xsd:anyType" abstract="true"/>
```
CityModelType, CityModel

```xml
<xs:complexType name="CityModelType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureCollectionType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

cityObjectMember

```xml
<xs:element name="cityObjectMember" type="gml:FeaturePropertyType" substitutionGroup="gml:featureMember"/>
```

AbstractSiteType, _Site

```xml
<xs:complexType name="AbstractSiteType" abstract="true">
  <xs:complexContent>
    <xs:extension base="AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfSite" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

10.1.2 Generalisation relation, RelativeToTerrainType and RelativeToWaterType

GeneralizationRelationType

```xml
<xs:complexType name="GeneralizationRelationType">
  <xs:sequence minOccurs="0">
    <xs:element ref="_CityObject"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```

RelativeToTerrainType, RelativeToWaterType

```xml
<xs:simpleType name="RelativeToTerrainType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="entirelyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveTerrain"/>
    <xs:enumeration value="substantiallyAboveAndBelowTerrain"/>
    <xs:enumeration value="substantiallyBelowTerrain"/>
    <xs:enumeration value="entirelyBelowTerrain"/>
  </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="RelativeToWaterType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="entirelyAboveWater"/>
    <xs:enumeration value="substantiallyAboveWater"/>
    <xs:enumeration value="substantiallyAboveAndBelowWater"/>
    <xs:enumeration value="substantiallyBelowWater"/>
    <xs:enumeration value="entirelyBelowWater"/>
  </xs:restriction>
</xs:simpleType>
```
10.1.3 External references

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

**ExternalReferenceType, ExternalObjectReferenceType**

```
<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>
```

10.1.4 Address information

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

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

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

**AddressPropertyType, AddressType, Address**

```
<xs:complexType name="AddressPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="Address"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
```
The following two excerpts of a CityGML dataset contain examples for the representation of German and British addresses in xAL. The address information is attached to building objects (<bldg:Building>) according to the CityGML Building module (cf. chapter 10.3). Generally, if a CityGML instance document contains address information, the namespace prefix “xAL” should be declared in the root element and must refer to “urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0”. An example showing a complete CityGML dataset including a building with an address element is provided in annex G.1.

```
<bldg:Building>
  ...
  <core:Address>
    <core:xalAddress>
      <!-- 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:ThoroughfareNumber>7</xAL:ThoroughfareNumber>
            <xAL:ThoroughfareName>Bussardweg</xAL:ThoroughfareName>
          </xAL:Thoroughfare>
          <xAL:PostalCode>
            <xAL:PostalCodeNumber>76356</xAL:PostalCodeNumber>
          </xAL:PostalCode>
        </xAL:Locality>
      </xAL:AddressDetails>
    </core:xalAddress>
  </core:Address>
</bldg:Building>  

<bldg:Building>
  ...
  <core:Address>
    <core:xalAddress>
      <!-- 46 Brynmaer Road Battersea LONDON, SW11 4EW United Kingdom -->
      <xAL:AddressDetails>
        <xAL:Country>
          <xAL:CountryName>United Kingdom</xAL:CountryName>
        </xAL:Country>
      </xAL:AddressDetails>
    </core:xalAddress>
  </core:Address>
</bldg:Building>
```
10.1.5 Code lists

The `mimeType` attribute of `ImplicitGeometry` is specified as `gml:CodeType`. The values of this property can be enumerated in a code list. A proposal for this code list can be found in annex C.6.

10.1.6 Conformance requirements

Base requirements

1. The `CityModel` element (type: `CityModelType`, substitutionGroup: `gml:_FeatureCollection`) shall only contain `cityObjectMember` elements (type: `gml:FeaturePropertyType`), `app:appearanceMember` elements (type: `app:AppearancePropertyType`), and `gml:featureMember` elements (type: `gml:FeaturePropertyType`) as feature members.

2. The type `ExternalObjectReference` introduces the two elements `name` (type: `xs:string`) and `uri` (type: `xs:anyURI`). The external reference may be specified by either of them. However, if the `informationSystem` property element (type: `xs:anyURI`) of the type `ExternalReferenceType` is not provided, the `uri` element of `ExternalObjectReference` must be given.

3. In order to represent address information about a feature, the corresponding thematic class of the feature shall define a property of the type `AddressPropertyType`. Thus, for all CityGML extension modules only the type `AddressPropertyType` shall be used for elements providing address information.

4. Since the concept of implicit geometries (cf. chapter 8.2) is part of the `CityGML Core` module, the conformance requirements introduced for implicit geometries (cf. chapter 8.3.3) are part of the conformance requirements of the core.

Referential integrity

5. The `cityObjectMember` element (type: `gml:FeaturePropertyType`) may contain a `CityObject` element, which typically is an object from a derived subclass like `bldg:Building`, inline or an XLink reference to a remote `CityObject` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `cityObjectMember` element may only point to a remote `CityObject` element (where remote `CityObject` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

6. The type `AddressPropertyType` may contain an `Address` element inline or an XLink reference to a remote `Address` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the corresponding element of type `AddressPropertyType` may only point to a remote `Address` element (where remote `Address` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.2 Digital Terrain Model (DTM)

An essential part of a city model is the terrain. The Digital Terrain Model (DTM) of CityGML is provided by the thematic extension module Relief (cf. chapter 7). In CityGML, the terrain is represented by the class ReliefFeature in LOD 0-4 (Fig. 24 depicts the UML diagram, for the XML schema definition see annex A.9). 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 ReliefFeature 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 gml:RectifiedGridCoverage, break lines by gml:MultiCurve, mass points by gml:MultiPoint and TINs either by gml:TriangulatedSurface or by gml:Tin. In case of gml:TriangulatedSurfaces, the triangles are given explicitly while in case of gml:Tin only 3D points are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation, cf. Okabe et al. 1992). Break lines are represented by 3D curves. Mass points are simply a set of 3D points.

In a CityGML dataset the four terrain types may be combined in different ways, yielding a high flexibility. First, each type may be represented in different levels of detail, reflecting different accuracies or resolutions. Second, a part of the terrain can be described by the combination of multiple types, for example by a raster and break lines, or by a TIN and break lines. In this case, the break lines must share the geometry with the triangles. Third, neighboring regions may be represented by different types of terrain models. To facilitate this combination, each terrain object is provided with a spatial attribute denoting its extent of validity (Fig. 25). In most cases, the extent of validity of a regular raster dataset corresponds to its bounding box. This validity extent is represented by a 2D footprint polygon, which may have holes. This concept enables, for example, the modelling of a terrain by a coarse grid, where some distinguished regions are represented by a detailed, high-accuracy TIN. The boundaries between both types are given by the extent attributes of the corresponding terrain objects.
Accuracy and resolution of the DTM are not necessarily dependent on features of other CityGML extension modules such as building models. Hence, there is the possibility to integrate building models with higher LOD to a DTM with lower accuracy or resolution.

This approach interacts with the concept of TerrainIntersectionCurves TIC (cf. chapter 6.5). The TIC can be used like break lines to adjust the DTM to different features such as buildings, bridges, or city furnitures, and hence to ensure a consistent representation of the DTM. If necessary, a retriangulation may have to be processed. A TIC can also be derived by the individual intersection of the DTM and the corresponding feature. ReliefFeature and its ReliefComponents both have an lod attribute denoting the corresponding level of detail. In most cases, the LOD of a ReliefFeature matches the LOD of its ReliefComponents. However, it is also allowed to specify a ReliefFeature with a high LOD which consists of ReliefComponents where some of them can have a LOD lower than that of the aggregating ReliefFeature. The idea is that, for example, for a LOD3 scene it might be sufficient to use a regular grid in LOD2 with certain higher precision areas defined by ReliefComponents in LOD3. The LOD2 grid and the LOD3 components can easily be integrated using the concept of the validity extent polygon. Therefore, although some of the ReliefComponents would have been classified to a lower LOD, the whole ReliefFeature would be appropriate to use with other LOD3 models which is indicated by setting its lod value to 3.

XML namespace

The XML namespace of the CityGML Relief module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/relief/2.0. Within the XML Schema definition of the Relief module, this URI is also used to identify the default namespace.

10.2.1 Relief feature and relief component

ReliefFeatureType, ReliefFeature

```
<xs:complexType name="ReliefFeatureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod" type="core:integerBetween0and4"/>
        <xs:element name="reliefComponent" type="ReliefComponentPropertyType" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfReliefFeature" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
AbstractReliefComponentType, _ReliefComponent

TINReliefType, TINRelief

10.2.3 Raster relief

RasterReliefType, RasterRelief, Elevation
10.2.4 Mass point relief

**MassPointReliefType, MassPointRelief**

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.
10.2.6 Conformance requirements

Base requirements

1. The gml:Polygon geometry element describing the extent of validity of a _ReliefComponent element using the extent property (type: gml:PolygonPropertyType) of _ReliefComponent shall be given as 2D footprint polygon which may have inner holes.

Referential integrity

2. The reliefComponent property (type: ReliefComponentPropertyType) of the element ReliefFeature may contain a _ReliefComponent element inline or an XLink reference to a remote _ReliefComponent element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the reliefComponent property may only point to a remote _ReliefComponent element (where remote _ReliefComponent elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

3. The tin property (type: tinPropertyType) of the element TINRelief may contain a gml:TriangulatedSurface element inline or an XLink reference to a remote gml:TriangulatedSurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the tin property may only point to a remote gml:TriangulatedSurface element (where remote gml:TriangulatedSurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

4. The grid property (type: gridPropertyType) of the element RasterRelief may contain a gml:RectifiedGridCoverage element inline or an XLink reference to a remote gml:RectifiedGridCoverage element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the grid property may only point to a remote gml:RectifiedGridCoverage element (where remote gml:RectifiedGridCoverage elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.3 Building model

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

Fig. 26: Examples for city or building models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left), and LOD4 (lower right) (source: District of Recklinghausen, m-g-h ingenieure+architekten GmbH).

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

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

Both classes Building and BuildingPart inherit the attributes of _AbstractBuilding: the class of the building, the function (e.g. residential, public, or industry), the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individual heights of the storeys above and below ground. This set of parameters is suited for roughly reconstructing the three-dimensional shape of a building and can be provided by cadastral systems. Furthermore, Address features can be assigned to Buildings or BuildingParts.
The geometric representation and semantic structure of an _AbstractBuilding_ is shown in Fig. 27. The model is successively refined from LOD0 to LOD4. Therefore, not all components of a building model are represented equally in each LOD and not all aggregation levels are allowed in each LOD. In CityGML, all object classes are associated to the LODs with respect to the proposed minimum acquisition criteria for each LOD (cf. chapter 6.2). An object can be represented simultaneously in different LODs by providing distinct geometries for the corresponding LODs.

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

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

In LOD3, the openings in _BoundarySurface objects (doors and windows) can be represented as thematic 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 (cf. chapter 10.3.6). Interior installations of a building, i.e. objects within a building which (in contrast to furniture) cannot be moved, are represented by the class IntBuildingInstallation. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the Room class, otherwise (e.g. in case of rafters or pipes) with AbstractBuilding. A Room may have the attributes class, function and usage whose value can be defined in code lists (chapter 10.3.8 and annex C.1). The class attribute allows a classification of rooms with respect to the stated function, e.g. commercial or private rooms, and occurs only once. The function attribute is intended to express the main purpose of the room, e.g. living room, kitchen. The attribute usage can be used if the way the object is actually used differs from the function. Both attributes can occur multiple times.

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

**XML namespace**

The XML namespace of the CityGML Building module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/building/2.0. Within the XML Schema definition of the Building module, this URI is also used to identify the default namespace.

### 10.3.1 Building and building part

**BuildingType, Building**

```xml
<xs:complexType name="BuildingType">
  <xs:complexContent>
    <xs:extension base="AbstractBuildingType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBuilding" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </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, for example the number of storeys or the roof type has to be separated into one `Building` having one or more additional `BuildingPart` (see Fig. 28). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating `Building` feature.

**BuildingPartType, BuildingPart**

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

**AbstractBuildingType, _AbstractBuilding**

Fig. 28: 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 in different levels of detail. The attributes describe:

a) The classification of the building or building part (class), the different intended usages (function), and the different actual usages (usage). The permitted values for these attributes can be specified in code lists.

b) The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the building or building part. These attributes can be used to describe the chronology of the building development within a city model. The points of time refer to real world time.

c) The roof type of the building or building part (roofType). The permitted values for this attribute can be specified in a code list.

d) The measured relative height (measuredHeight) of the building or building part.

e) The number of storeys above (storeyAboveGround) and below (storeyBelowGround) ground level.

f) 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. 29 and Fig. 30, showing the same building in five different LODs. The class _AbstractBuilding_ has a number of properties which are associated with certain LODs.
Fig. 29: The two possibilities of modeling a building in LOD0 using horizontal 3D surfaces. On the left, the building footprint (lod0FootPrint) is shown (cyan) which denotes the shape of the building on the ground. The corresponding surface representation is located at ground level. On the right, the lod0RoofEdge representation is illustrated (cyan) which results from a horizontal projection of the building’s roof and which is located at the eaves height (source: Karlsruhe Institute of Technology (KIT), courtesy of Franz-Josef Kaiser).

Fig. 30: Building model in LOD1 – LOD4 (source: Karlsruhe Institute of Technology (KIT), courtesy of Franz-Josef Kaiser).

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

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

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

In LOD2 and higher levels of detail, the exterior shell of a building is not only represented geometrically as gml:Solid geometry and/or a gml:MultiSurface geometry, but it can also be composed of semantic objects. The base class for all objects semantically structuring the building shell is _BoundarySurface (cf. chapter 10.3.2), which is associated with a gml:MultiSurface geometry. If in a building model there is both a geometric representation of the exterior shell as volume or surface model and a semantic representation by means of thematic _BoundarySurfaces, the geometric representation must not explicitly define the geometry, but has to reference the corresponding geometry components of the gml:MultiSurface of the _BoundarySurface elements.
Apart from BuildingParts, smaller features of the building (“outer building installations”) can also strongly affect the building characteristic. These features are modelled by the class BuildingInstallation (cf. chapter 10.3.2). Typical candidates for this class are chimneys (see Fig. 30), dormers (see Fig. 28), balconies, outer stairs, or antennas. BuildingInstallations may only be included in LOD2 models, if their extents exceed the proposed minimum dimensions as specified in chapter 6.2. For the geometrical representation of the class BuildingInstallation, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used.

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

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

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

### 10.3.2 Outer building installations

**BuildingInstallationType, BuildingInstallation**

<table>
<thead>
<tr>
<th>Geometric / semantic theme</th>
<th>Property type</th>
<th>LOD0</th>
<th>LOD1</th>
<th>LOD2</th>
<th>LOD3</th>
<th>LOD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building footprint and roof edge</td>
<td>gml:MultiSurfaceType</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volume part of the building shell</td>
<td>gml:SolidType</td>
<td>•</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>
<td></td>
</tr>
<tr>
<td>Terrain intersection curve</td>
<td>gml:MultiCurveType</td>
<td>•</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>
<td></td>
</tr>
<tr>
<td>Building parts</td>
<td>BuildingPartType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Boundary surfaces (chapter 10.3.3)</td>
<td>AbstractBoundarySurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Outer building installations (chapter 10.3.2)</td>
<td>BuildingInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Openings (chapter 10.3.4)</td>
<td>AbstractOpeningType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Rooms (chapter 10.3.5)</td>
<td>RoomType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
<tr>
<td>Interior building installations (chapter 10.3.5)</td>
<td>IntBuildingInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
</tr>
</tbody>
</table>

Tab. 5: Semantic themes of the class _AbstractBuilding.

---

**Copyright © 2012 Open Geospatial Consortium.**
A *BuildingInstallation* is an outer component of a building which has not the significance of a *BuildingPart*, but which strongly affects the outer characteristic of the building. Examples are chimneys, stairs, antennas, balconies or attached roofs above stairs and paths. A *BuildingInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class* - which can only occur once - represents a general classification of the installation. With the attributes *function* and *usage*, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of a *BuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype building installation is stored only once in a local coordinate system and referenced by other building installation features (see chapter 8.2). The visible surfaces of a building installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3). A *BuildingInstallation* object should be uniquely related to exactly one building or building part object.

### 10.3.3 Boundary surfaces

*AbstractBoundarySurfaceType, _BoundarySurface*

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

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

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

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

OuterCeilingSurfaceType, OuterCeilingSurface

The ground plate of a building or building part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the building’s footprint. However, the surface normal of the ground plate is pointing downwards.
A mostly horizontal surface belonging to the outer building shell and having the orientation pointing downwards can be modeled as an *OuterCeilingSurface*. Examples are the visible part of the ceiling of a loggia or the ceiling of a passage.

**WallSurfaceType, WallSurface**

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

**OuterFloorSurfaceType, OuterFloorSurface**

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

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

**ClosureSurfaceType, ClosureSurface**

```xml
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

**FloorSurfaceType, FloorSurface**

```xml
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

**InteriorWallSurfaceType, InteriorWallSurface**

```xml
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

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

10.3.4 Openings

AbstractOpeningType, _Opening

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

WindowType, Window

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

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

10.3.5 Building interior

RoomType, Room

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

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

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

![Accessibility graph](image)

Fig. 33: Accessibility graph derived from topological adjacencies of room surfaces (graphic: IGG Uni Bonn).

**BuildingFurnitureType, BuildingFurniture**

```xml
<xs:complexType name="BuildingFurnitureType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfBuildingFurniture" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Rooms may have **BuildingFurnitures** and **IntBuildingInstallations**. A **BuildingFurniture** is a movable part of a room, such as a chair or furniture. A **BuildingFurniture** object should be uniquely related to exactly one room object. Its geometry may be represented by an explicit geometry or an **ImplicitGeometry** object. Following the concept of **ImplicitGeometry** the geometry of a prototype building furniture is stored only once in a local coordinate system and referenced by other building furniture features (see chapter 8.2).

**IntBuildingInstallationType, IntBuildingInstallation**

```xml
<xs:complexType name="IntBuildingInstallationType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfIntBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
An *IntBuildingInstallation* is an object inside a building with a specialised function or semantic meaning. In contrast to *BuildingFurniture*, *IntBuildingInstallations* are permanently attached to the building structure and cannot be moved. Typical examples are interior stairs, railings, radiators or pipes. Objects of the class *IntBuildingInstallation* can either be associated with a room (class *Room*), or with the complete building / building part (class *AbstractBuilding*, cf. chapter 10.3.1). However, they should be uniquely related to exactly one room or one building / building part object. An *IntBuildingInstallation* optionally has attributes *class*, *function* and *usage*. The attribute *class*, which can only occur once, represents a general classification of the internal building component. With the attributes *function* and *usage*, nominal and real functions of a building installation can be described. For all three attributes the list of feasible values can be specified in a code list. For the geometrical representation of an *IntBuildingInstallation*, an arbitrary geometry object from the GML subset shown in Fig. 9 can be used. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype interior building installation is stored only once in a local coordinate system and referenced by other interior building installation features (see chapter 8.2). The visible surfaces of an interior building installation can be semantically classified using the concept of boundary surfaces (cf. 10.3.3).

10.3.6 Modelling building storeys using *CityObjectGroups*

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

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

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

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

10.3.7 Examples

The LOD1 model of the Campus North of the Karlsruhe Institute of Technology (KIT) shown in Fig. 34 consists of 596 buildings and 187 building parts. The footprint geometries of the buildings are taken from a cadastral information system and extruded by a given height. Buildings with a unique identifier and a single height value are modeled as one building (*bldg:Building*). Buildings having a unique identifier but different height values are modeled as one building (*bldg:Building*) with one or more building parts (*bldg:BuildingPart*). Both buildings and building parts have solid geometries and their height values are additionally represented as thematic attribute (*bldg:measuredHeight*). Fig. 34 shows an aerial photograph of the KIT Campus North (left) and the CityGML LOD1 model (right).
An example for a fully textured LOD2 building model is given in Fig. 35 which shows the Bernhardus church located in the city of Karlsruhe, Germany. On the left side of Fig. 35, a photograph of the church in real world is shown whereas the CityGML building model of the church with photorealistic textures is illustrated on the right. The model is bounded by a ground surface, several wall and roof surfaces. The railing above the church clock is modeled as a building installation (BuildingInstallation).

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

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

The attributes class, function, usage, and roofType of the feature _AbstractBuilding as well as the attributes class, function and usage of the features BuildingInstallation, Room, BuildingFurniture and IntBuildingInstallation are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.1.

10.3.9 Conformance requirements

Base requirements

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

Usage restriction of building model components according to different LODs

2. The gml:MultiSurface geometries that are associated using the lod0FootPrint and lod0RoofEdge properties must have 3D coordinates. For each surface, the height values of the coordinate tuples belonging to the same surface shall be identical.

3. The lodXSolid and lodXMultiSurface, X ∈ [1..4], properties (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType) of _AbstractBuilding may be used to geometrically represent the exterior shell of a building (as volume or surface model) within each LOD. For LOD1, either lod1Solid or lod1MultiSurface must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.

4. Starting from LOD2, the exterior shell of an _AbstractBuilding may be semantically decomposed into _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of _AbstractBuilding. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of _BoundarySurface are allowed. The boundedBy property (not to be confused with the gml:boundedBy property) shall not be used if the building is only represented in LOD1.

If the exterior shell is represented by _BoundarySurface elements, an additional geometric representation as volume or surface model using the lodXSolid and lodXMultiSurface, X ∈ [2..4], properties shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of _BoundarySurface within each LOD using the XLink concept of GML 3.1.1.

5. Starting from LOD2, curve parts of the building shell may be represented using the lodXMultiCurve, X ∈ [2..4], property of _AbstractBuilding. This property shall not be used if the building is only represented in LOD1.

6. Starting from LOD2, the outerBuildingInstallation property (type: BuildingInstallationPropertyType) of _AbstractBuilding may be used to model BuildingInstallation elements. BuildingInstallation elements shall only be used to represent outer characteristics of a building which do not have the significance of...
building parts. The outerBuildingInstallation property shall not be used if the building is only represented in LOD 1.

7. Starting from LOD2, the geometry of BuildingInstallation elements may be semantically classified by BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of BuildingInstallation. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of BoundarySurface are allowed.

8. Starting from LOD3, openings of BoundarySurface elements may be modelled using the opening property (type: OpeningPropertyType) of BoundarySurface. This property shall not be used for BoundarySurface elements only represented in LOD2. Accordingly, the surface geometry representing a BoundarySurface in LOD2 must be simply connected.

The opening property of BoundarySurface may contain or reference Opening elements. If the geometric location of an Opening element topologically lies within a surface component of the BoundarySurface, the opening must also be represented as inner hole of that surface. The embrasure surface of an Opening element shall belong to the relevant adjacent BoundarySurface.

9. Starting from LOD4, the interiorRoom property (type: InteriorRoomPropertyType) of AbstractBuilding may be used to semantically model the free space inside the building by Room elements. This property shall not be used if the building is only represented in LOD 1 – 3. The Room element may be geometrically represented as a surface or volume model, using its lod4Solid or lod4MultiSurface property (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType).

In addition, different parts of the visible surface of a room may be modelled by thematic BoundarySurface elements. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of BoundarySurface are allowed. If the visible surface of a room is represented by BoundarySurface elements, an additional geometric representation as volume or surface model using the lod4Solid and lod4MultiSurface property shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of BoundarySurface using the XLink concept of GML 3.1.1.

10. Starting from LOD4, the interiorBuildingInstallation property (type: IntBuildingInstallationPropertyType) of AbstractBuilding may be used to represent immovable objects inside the building that are permanently attached to the building structure. The interiorBuildingInstallation property shall not be used if the building is only represented in LOD 1 – 3. Furthermore, the interiorBuildingInstallation property shall only be used if the object cannot be associated with a Room element. In the latter case, the roomInstallation property (type: IntBuildingInstallationPropertyType) of the corresponding Room element shall be used to represent the object.

11. Starting from LOD4, the geometry of IntBuildingInstallation elements may be semantically classified by BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of IntBuildingInstallation. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of BoundarySurface are allowed.

Referential integrity

12. The boundedBy property (type: BoundarySurfacePropertyType) of the element AbstractBuilding may contain a BoundarySurface element inline or an XLink reference to a remote BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote BoundarySurface element (where remote BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of AbstractBuilding.

13. The outerBuildingInstallation property (type: BuildingInstallationPropertyType) of the element AbstractBuilding may contain a BuildingInstallation element inline or an XLink reference to a remote BuildingInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the outerBuildingInstallation property may only point to a remote BuildingInstallation element (where remote BuildingInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
14. The `interiorBuildingInstallation` property (type: `IntBuildingInstallationPropertyType`) of the element `_AbstractBuilding` may contain an `IntBuildingInstallation` element inline or an XLink reference to a remote `IntBuildingInstallation` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `interiorBuildingInstallation` property may only point to a remote `IntBuildingInstallation` element (where remote `IntBuildingInstallation` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

15. The `interiorRoom` property (type: `InteriorRoomPropertyType`) of the element `_AbstractBuilding` may contain a `Room` element inline or an XLink reference to a remote `Room` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `interiorRoom` property may only point to a remote `Room` element (where remote `Room` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

16. The `consistsOfBuildingPart` property (type: `BuildingPartPropertyType`) of the element `_AbstractBuilding` may contain a `BuildingPart` element inline or an XLink reference to a remote `BuildingPart` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `consistsOfBuildingPart` property may only point to a remote `BuildingPart` element (where remote `BuildingPart` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

17. The `address` property (type: `core:AddressPropertyType`) of the element `_AbstractBuilding` may contain an `core:Address` element inline or an XLink reference to a remote `core:Address` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `address` property may only point to a remote `core:Address` element (where remote `core:Address` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

18. The `opening` property (type: `OpeningPropertyType`) of the element `_BoundarySurface` may contain an `_Opening` element inline or an XLink reference to a remote `_Opening` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `opening` property may only point to a remote `_Opening` element (where remote `_Opening` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

19. The `address` property (type: `core:AddressPropertyType`) of the element `Door` may contain an `core:Address` element inline or an XLink reference to a remote `core:Address` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `address` property may only point to a remote `core:Address` element (where remote `core:Address` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

20. The `boundedBy` property (type: `BoundarySurfacePropertyType`) of the element `BuildingInstallation` may contain a `_BoundarySurface` element inline or an XLink reference to a remote `_BoundarySurface` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `boundedBy` property may only point to a remote `_BoundarySurface` element (where remote `_BoundarySurface` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only `RoofSurface`, `WallSurface`, `GroundSurface`, `OuterCeilingSurface`, `OuterFloorSurface` and `ClosureSurface` elements are allowed to be encapsulated or referenced by the `boundedBy` property of `BuildingInstallation`.

21. The `boundedBy` property (type: `BoundarySurfacePropertyType`) of the element `IntBuildingInstallation` may contain a `_BoundarySurface` element inline or an XLink reference to a remote `_BoundarySurface` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `boundedBy` property may only point to a remote `_BoundarySurface` element (where remote `_BoundarySurface` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of IntBuildingInstallation.

22. The boundedBy property (type: BoundarySurfacePropertyType) of the element Room may contain a BoundarySurface element inline or an XLink reference to a remote BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote BoundarySurface element (where remote BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of Room.

23. The interiorFurniture property (type: InteriorFurniturePropertyType) of the element Room may contain a BuildingFurniture element inline or an XLink reference to a remote BuildingFurniture element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorFurniture property may only point to a remote BuildingFurniture element (where remote BuildingFurniture elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

24. The roomInstallation property (type: IntBuildingInstallationPropertyType) of the element Room may contain an IntBuildingInstallation element inline or an XLink reference to a remote IntBuildingInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the roomInstallation property may only point to a remote IntBuildingInstallation element (where remote IntBuildingInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

25. The lodXImplicitRepresentation, $X \in [2..4]$, property (type: core:ImplicitRepresentationPropertyType) of the element BuildingInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, $X \in [2..4]$, property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

26. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element IntBuildingInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

27. The lodXImplicitRepresentation, $X \in [3..4]$, property (type: core:ImplicitRepresentationPropertyType) of the element _Opening may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, $X \in [3..4]$, property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

28. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element BuildingFurniture may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.4 Tunnel model

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

![Tunnel models examples](image)

**Fig. 37:** Examples for tunnel models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left) and LOD4 (lower right) (source: Google 3D warehouse).

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

Both classes _Tunnel_ and _TunnelPart_ inherit the attributes of _AbstractTunnel_: the class of the tunnel, the function, the usage, the year of construction and the year of demolition. In contrast to _AbstractBuilding_, _Address_ features cannot be assigned to _AbstractTunnel_.

---

82 Copyright © 2012 Open Geospatial Consortium.
D. In CityGML, all object classes are `lod3MultiSurface`. This would require the precise definition of a "transition surface".

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

Similar to the building and bridge models (cf. chapters 10.3 and 10.5), only the outer shell of a tunnel is represented in LOD – 3, which is composed of the tunnel’s boundary surfaces to the surrounding earth, water, or outdoor air. The interior of a tunnel may only be modeled in LOD4. Although the interior built environment is especially relevant for subsurface objects like tunnels or underground buildings, CityGML employs a consistent LOD concept for all thematic modules. If, in contrast, the representation of the interior of subsurface objects would be possible in all LODs, the LOD concept for subsurface objects would have to substantially differ from the LOD concept for aboveground objects. This would require the precise definition of a "transition surface" which delimits the scope of both LOD concepts. Furthermore, features being partially above and below ground would have to be split into an above-ground part (modeled according to the aboveground LOD concept) and a

Fig. 38: UML diagram of CityGML’s tunnel model. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Tunnel module.
subsurface part (modeled according to the subsurface LOD concept). However, such a splitting violates the CityGML concept of unity of features and would not be feasible in many cases where the transition between above and below ground is often not precisely known or depends on (the LOD of) the terrain model. Hence, CityGML applies a single and consistent LOD concept to both aboveground and subsurface objects. As a consequence, penetrations between a tunnel and objects inside this tunnel (e.g., roads and railways) may occur in LOD1 – 3.

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

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

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

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

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

XML namespace

The XML namespace of the CityGML Tunnel module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/tunnel/2.0. Within the XML Schema definition of the Tunnel module, this URI is also used to identify the default namespace.

10.4.1 Tunnel and tunnel part

TunnelType, Tunnel

```xml
<xs:complexType name="TunnelType">
  <xs:complexContent>
    <xs:extension base="AbstractTunnelType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfTunnel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Copyright © 2012 Open Geospatial Consortium.
The *Tunnel* class is one of the two subclasses of *AbstractTunnel*. If a tunnel only consists of one (homogeneous) part, this class shall be used. A tunnel composed of structural segments, for example tunnel entrance and subway, has to be separated into one tunnel having one or more additional *TunnelPart* (see Fig. 39). The geometry and non-spatial properties of the central part of the tunnel should be represented in the aggregating *Tunnel* feature.

**TunnelPartType, TunnelPart**

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

**AbstractTunnelType, _AbstractTunnel**
This is illustrated in Fig. 40, showing the same tunnel in four different LODs. Some properties of the class _AbstractTunnel_ are also associated with certain LODs.

![Diagram of LODs](image)

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

a) The classification of the tunnel or tunnel part (class), the different functions (function), and the usage (usage). The type of these attributes is gml:CodeType and the values can be specified in separate code lists.

b) The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the tunnel or tunnel part. The yearOfConstruction is the year of completion of the tunnel. The yearOfDemolition is the year when the demolition of the tunnel was completed. The date (year) refer to real world time (e.g. 2011).

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

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

<table>
<thead>
<tr>
<th>Geometric / semantic theme</th>
<th>Property type</th>
<th>LOD1</th>
<th>LOD2</th>
<th>LOD3</th>
<th>LOD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume part of the tunnel shell</td>
<td>gml:SolidType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Surface part of the tunnel 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 tunnel shell</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Tunnel parts</td>
<td>TunnelPartType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Boundary surfaces (chapter 10.4.3)</td>
<td>AbstractBoundarySurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Outer tunnel installations (chapter 10.4.2)</td>
<td>TunnelInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Openings</td>
<td>AbstractOpeningType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Hollow spaces (chapter 10.4.5)</td>
<td>HollowSpaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Interior tunnel installations</td>
<td>IntTunnelInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

Tab. 6: Semantic themes of the class _AbstractTunnel.

10.4.2 Outer tunnel installations

TunnelInstallationType, TunnelInstallation

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

10.4.3 Boundary surfaces

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

![Diagram of CityGML specific properties](image)

**Fig. 41**: Examples for the use of boundary surfaces for tunnels with different cross sections. `WallSurface`, `RoofSurface`, `GroundSurface`, `OuterCeilingSurface` and `OuterFloorSurface` are available in LOD2 – 4, whereas `InteriorWallSurface`, `FloorSurface` and `CeilingSurface` may only be used in LOD4 to model the interior boundary surfaces of a hollow space.

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

**GroundSurfaceType, GroundSurface**

```xml
<xs:complexType name="GroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
The ground plate of a tunnel or tunnel part is modelled by the class `GroundSurface`. Usually a `GroundSurface` is a boundary surface between the tunnel and the surrounding earth (soil, rock etc.) or water.

**OuterCeilingSurfaceType, OuterCeilingSurface**

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

**WallSurfaceType, WallSurface**

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

**OuterFloorSurfaceType, OuterFloorSurface**

A mostly horizontal surface belonging to the outer tunnel shell and with the orientation pointing upwards can be modeled as an `OuterFloorSurface`. 
Boundary surfaces belonging to the outer tunnel shell and with the main purpose to protect the tunnel from above are expressed by the class **RoofSurface**. The orientation of these boundaries is mainly pointing upwards.

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

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

---

**RoofSurfaceType, RoofSurface**

```xml
<xs:complexType name="RoofSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

**ClosureSurfaceType, ClosureSurface**

```xml
<xs:complexType name="ClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

**FloorSurfaceType, FloorSurface**

```xml
<xs:complexType name="FloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

**InteriorWallSurfaceType, InteriorWallSurface**

```xml
<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
The class `InteriorWallSurface` is only allowed to be used in the LOD4 interior tunnel model for modelling the visible wall surfaces of hollow spaces.

**CeilingSurfaceType, CeilingSurface**

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

### 10.4.4 Openings

**AbstractOpeningType, _Opening**

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

**WindowType, Window**

The class `Window` is only allowed to be used in the LOD4 interior tunnel model for modelling windows of hollow spaces.
The class Window is used for modelling windows in the in the exterior shell of a tunnel and in hollow spaces, or
hatches between adjacent hollow spaces. The formal difference between the classes Window and Door is that –
in normal cases – Windows are not specifically intended for the transit of people or vehicles.

**DoorType, Door**

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

### 10.4.5 Tunnel interior

**HollowSpaceType, HollowSpace**

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

In addition to the geometrical representation, different parts of the visible surface of a hollow space can be
modelled by specialised boundary surfaces (FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSur-
face, cf. chapter 10.4.3).
Hollow spaces may have **TunnelFurniture**. A **TunnelFurniture** is a movable part of a hollow space. A **TunnelFurniture** object should be uniquely related to exactly one hollow space. Its geometry may be represented by an explicit geometry or an **ImplicitGeometry** object. Following the concept of **ImplicitGeometry** the geometry of a prototype tunnel furniture is stored only once in a local coordinate system and referenced by other tunnel furniture features (see chapter 8.2).

**IntTunnelInstallationType, IntTunnelInstallation**

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

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

![Fig. 42: Example of a tunnel modeled in LOD3 (real situation on the left side; CityGML model on the right side) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).](image)

![Fig. 43: The same LOD3 tunnel shown from a different perspective. The camera is positioned in front of the left entrance and pointing in the direction of the right entrance. (real situation on the left side; CityGML model on the right side). The model on the right also includes an LOD1 representation of the nearby buildings in the background (painted in light brown) (source: Karlsruhe Institute of Technology (KIT), courtesy of City of Karlsruhe).](image)

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

10.4.7 Code lists

The attributes *class*, *function*, and *usage* of the features *AbstractTunnel*, *TunnelInstallation*, *HollowSpace*, *TunnelFurniture* and *IntTunnelInstallation* are specified as *gml:CodeType*. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.2.
10.4.8 Conformance requirements

Base requirements

1. If a tunnel only consists of one (homogeneous) part, it shall be represented by the element *Tunnel*. However, if a tunnel is composed of individual structural segments, it shall be modelled as a *Tunnel* element having one or more additional *TunnelPart* elements. Only the geometry and non-spatial properties of the main part of the tunnel should be represented within the aggregating *Tunnel* element.

Usage restriction of tunnel model components according to different LODs

2. The *lodXSolid* and *lodXMultiSurface*, \( X \in [1..4] \), properties (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType) of *AbstractTunnel* may be used to geometrically represent the exterior shell of a tunnel (as volume or surface model) within each LOD. For LOD1, either *lod1Solid* or *lod1MultiSurface* must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.

3. Starting from LOD2, the exterior shell of an *AbstractTunnel* may be semantically decomposed into *BoundarySurface* elements using the boundedBy property (type: *BoundarySurfacePropertyType*) of *AbstractTunnel*. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of *BoundarySurface* are allowed. The boundedBy property (not to be confused with the gml:boundedBy property) shall not be used if the tunnel is only represented in LOD1.

   If the exterior shell is represented by *BoundarySurface* elements, an additional geometric representation as volume or surface model using the *lodXSolid* and *lodXMultiSurface*, \( X \in [2..4] \), properties shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of *BoundarySurface* within each LOD using the XLink concept of GML 3.1.1.

4. Starting from LOD2, curve parts of the tunnel shell may be represented using the *lodXMultiCurve*, \( X \in [2..4] \), property of *AbstractTunnel*. This property shall not be used if the tunnel is only represented in LOD1.

5. Starting from LOD2, the *outerTunnelInstallation* property (type: *TunnelInstallationPropertyType*) of *AbstractTunnel* may be used to model *TunnelInstallation* elements. *TunnelInstallation* elements shall only be used to represent outer characteristics of a tunnel which do not have the significance of tunnel parts. The *outerTunnelInstallation* property shall not be used if the tunnel is only represented in LOD1.

6. Starting from LOD2, the geometry of *TunnelInstallation* elements may be semantically classified by *BoundarySurface* elements using the boundedBy property (type: *BoundarySurfacePropertyType*) of *TunnelInstallation*. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of *BoundarySurface* are allowed.

7. Starting from LOD3, openings of *BoundarySurface* elements may be modelled using the opening property (type: *OpeningPropertyType*) of *BoundarySurface*. This property shall not be used for *BoundarySurface* elements only represented in LOD2. Accordingly, the surface geometry representing a *BoundarySurface* in LOD2 must be simply connected.

   The opening property of *BoundarySurface* may contain or reference *Opening* elements. If the geometric location of an *Opening* element topologically lies within a surface component of the *BoundarySurface*, the opening must also be represented as inner hole of that surface. The embrasure surface of an *Opening* element shall belong to the relevant adjacent *BoundarySurface*.

8. Starting from LOD4, the *interiorHollowSpace* property (type: *InteriorHollowSpacePropertyType*) of *AbstractTunnel* may be used to semantically model the free space inside the tunnel by *HollowSpace* elements. This property shall not be used if the tunnel is only represented in LOD 1 – 3. The *HollowSpace* element may be geometrically represented as a surface or volume model, using its *lod4Solid* or *lod4MultiSurface* property (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType).

In addition, different parts of the visible surface of a hollow space may be modelled by thematic *BoundarySurface* elements. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of *BoundarySurface* are allowed. If the visible surface of a hollow space is repre-
sented by _BoundarySurface elements, an additional geometric representation as volume or surface model using the lod4Solid and lod4MultiSurface property shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of _BoundarySurface using the XLink concept of GML 3.1.1.

9. Starting from LOD4, the interiorTunnelInstallation property (type: IntTunnelInstallationPropertyType) of _AbstractTunnel may be used to represent immovable objects inside the tunnel that are permanently attached to the tunnel structure. The interiorTunnelInstallation property shall not be used if the tunnel is only represented in LOD 1 – 3. Furthermore, the interiorTunnelInstallation property shall only be used if the object cannot be associated with a HollowSpace element. In the latter case, the hollowSpaceInstallation property (type: IntTunnelInstallationPropertyType) of the corresponding HollowSpace element shall be used to represent the object.

10. Starting from LOD4, the geometry of IntTunnelInstallation elements may be semantically classified by _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of IntTunnelInstallation. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of _BoundarySurface are allowed.

Referential integrity

11. The boundedBy property (type: BoundarySurfacePropertyType) of the element _AbstractTunnel may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of _AbstractTunnel.

12. The outerTunnelInstallation property (type: TunnelInstallationPropertyType) of the element _AbstractTunnel may contain a TunnelInstallation element inline or an XLink reference to a remote TunnelInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the outerTunnelInstallation property may only point to a remote TunnelInstallation element (where remote TunnelInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

13. The interiorTunnelInstallation property (type: IntTunnelInstallationPropertyType) of the element _AbstractTunnel may contain an IntTunnelInstallation element inline or an XLink reference to a remote IntTunnelInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorTunnelInstallation property may only point to a remote IntTunnelInstallation element (where remote IntTunnelInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

14. The interiorHollowSpace property (type: InteriorHollowSpacePropertyType) of the element _AbstractTunnel may contain a HollowSpace element inline or an XLink reference to a remote HollowSpace element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorHollowSpace property may only point to a remote HollowSpace element (where remote HollowSpace elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

15. The consistsOfTunnelPart property (type: TunnelPartPropertyType) of the element _AbstractTunnel may contain a TunnelPart element inline or an XLink reference to a remote TunnelPart element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the consistsOfTunnelPart property may only point to a remote TunnelPart element (where remote TunnelPart elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

16. The opening property (type: OpeningPropertyType) of the element _BoundarySurface may contain an _Opening element inline or an XLink reference to a remote _Opening element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the opening property may only point to a re-
mote _Opening element (where remote _Opening elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

17. The boundedBy property (type: BoundarySurfacePropertyType) of the element TunnelInstallation may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of TunnelInstallation.

18. The boundedBy property (type: BoundarySurfacePropertyType) of the element IntTunnelInstallation may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of IntTunnelInstallation.

19. The boundedBy property (type: BoundarySurfacePropertyType) of the element HollowSpace may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of HollowSpace.

20. The interiorFurniture property (type: InteriorFurniturePropertyType) of the element HollowSpace may contain a TunnelFurniture element inline or an XLink reference to a remote TunnelFurniture element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorFurniture property may only point to a remote TunnelFurniture element (where remote TunnelFurniture elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

21. The hollowSpaceInstallation property (type: IntTunnelInstallationPropertyType) of the element HollowSpace may contain an IntTunnelInstallation element inline or an XLink reference to a remote IntTunnelInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the hollowSpaceInstallation property may only point to a remote IntTunnelInstallation element (where remote IntTunnelInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

22. The lodXImplicitRepresentation, \(X \in [2..4]\), property (type: core:ImplicitRepresentationPropertyType) of the element TunnelInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, \(X \in [2..4]\), property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

23. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element IntTunnelInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote
core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

24. The lodXImplicitRepresentation, X ∈ [3..4], property (type: core:ImplicitRepresentationPropertyType) of the element _Opening may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, X ∈ [3..4], property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

25. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element TunnelFurniture may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.5 Bridge model

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

Fig. 44: Examples for bridge models in LOD1 (upper left), LOD2 (upper right), LOD3 (lower left) and LOD4 (lower right) (source: Google 3D warehouse)

The bridge model was developed in analogy to the building model (cf. chapter 10.3) with regard to structure and attributes. The UML diagram of the bridge model is depicted in Fig. 45, and the XML schema definition is presented in annex A.3.
A (movable or unmovable) bridge is represented by an object of the class _AbstractBridge_. This class inherits its attributes and relations from the abstract base class _AbstractBridge_. The spatial properties are defined by a solid for each of the four LODs (relations _lod1Solid to _lod4Solid_). In analogy to the building model, the semantical as well as the geometrical richness increases from LOD1 (blocks model) to LOD3 (architectural model). Simple examples of bridges in each of those LODs are depicted in Fig. 46. Interior structures like rooms are dedicated to LOD4. To cover the case of bridge models where the topology does not satisfy the properties of a solid (essentially water tightness), a multi surface representation is allowed (_lod1MultiSurface to _lod4MultiSurface_). The line where the bridge touches the terrain surface is represented by a terrain intersection curve, which is provided for each LOD (relations _lod1TerrainIntersection to _lod4TerrainIntersection_). In addition to the solid representation of a bridge, linear characteristics like ropes or antennas can be specified geometrically by the _lod1MultiCurve to _lod4MultiCurve_ relations. If those characteristics shall be represented semantically, the features BridgeInstallation or BridgeConstructionElement can be used (see section 10.5.2). All relations to semantic objects and geometric properties are listed in Tab. 7.

Fig. 45: UML diagram of the bridge model, part one.
The semantic attributes of an _AbstractBridge are class, function, usage and is_movable. The attribute class is used to classify bridges, e.g. to distinguish different construction types (cf. Fig. 48). The attribute function allows representing the utilization of the bridge independently of the construction. Possible values may be railway bridge, roadway bridge, pedestrian bridge, aqueduct, etc. The option to denote a usage which is divergent to one of the primary functions of the bridge (function) is given by the attribute usage. The type of these attributes is gml:CodeType, the values of which can be defined in code lists. The name of the bridge can be represented by the gml:name attribute, which is inherited from the base class gml:_GML via the classes gml:_Feature, _CityObject, and _Site. Each Bridge or BridgePart feature may be assigned zero or more addresses using the address property. The corresponding AddressPropertyType is defined within the CityGML core module (cf. chapter 10.1.4).

<table>
<thead>
<tr>
<th>Geometric / semantic theme</th>
<th>Property type</th>
<th>LOD1</th>
<th>LOD2</th>
<th>LOD3</th>
<th>LOD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume part of the bridge shell</td>
<td>gml:SolidType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Surface part of the bridge 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 bridge shell</td>
<td>gml:MultiCurveType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Bridge parts (chapter 10.5.1)</td>
<td>BridgePartType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Boundary surfaces (chapter 10.5.3)</td>
<td>AbstractBoundarySurfaceType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Outer bridge installations (chapter 10.5.2)</td>
<td>BridgeInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Bridge construction elements (chapter 10.5.2)</td>
<td>BridgeConstruction-ElementType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Openings (chapter 10.5.4)</td>
<td>AbstractOpeningType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Bridge rooms (chapter 10.5.5)</td>
<td>BridgeRoomType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Interior bridge installations</td>
<td>IntBridgeInstallationType</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

Tab. 7: Semantic themes of the class _AbstractBridge.
The boolean attribute \textit{is\_movable} is defined to specify whether a bridge is movable or not. The modeling of the geometric aspects of the movement is delayed to later versions of this standard. Some types of movable bridges are depicted in Fig. 47.

![Fig. 47: Examples for movable bridges (source: ISO 6707).](image)

Some types of movable bridges are depicted in Fig. 47.

![Fig. 47: Examples for movable bridges (source: ISO 6707).](image)

![Fig. 47: Examples for movable bridges (source: ISO 6707).](image)

![Fig. 47: Examples for movable bridges (source: ISO 6707).](image)

XML namespace

The XML namespace of the CityGML Bridge module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/bridge/2.0. Within the XML Schema definition of the Bridge module, this URI is also used to identify the default namespace.

10.5.1 Bridge and bridge part

\textbf{BridgeType, Bridge}

```xml
<xs:complexType name="BridgeType">  
  <xs:complexContent>  
    <xs:extension base="AbstractBridgeType">  
      <xs:sequence>  
        <xs:element ref="_GenericApplicationPropertyOfBridge" minOccurs="0" maxOccurs="unbounded"/>  
      </xs:sequence>  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>
```
BridgePartType, BridgePart

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

AbstractBridgeType, _AbstractBridge

Copyright © 2012 Open Geospatial Consortium.
The abstract class _AbstractBridge is the base class of Bridges and BridgeParts. It contains properties for bridge attributes, purely geometric representations, and geometric/semantic representations of the bridge or bridge part in different levels of detail. The attributes describe:

a) The classification of the bridge or bridge part (class), the different intended usages (function), and the different actual usages (usage). The permitted values for these property types can be specified in code lists.

b) The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the bridge or bridge part. These attributes can be used to describe the chronology of the bridge development within a city model. The points of time refer to real world time.

c) Whether the bridge is movable is specified by the Boolean attribute isMovable.

10.5.2 Bridge construction elements and bridge installations

BridgeConstructionElementType, BridgeConstructionElement

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

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

![Fig. 49: BridgeConstructionElements of a suspension bridge.](image-url)
10.5.3 Boundary surfaces

AbstractBoundarySurfaceType, _BoundarySurface

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

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

In LOD3 and LOD4, a _BoundarySurface may contain _Openings (cf. chapter 10.5.4) like doors and windows. If the geometric location of _Openings topologically lies within a surface component (e.g. gml:Polygon) of the gml:MultiSurface geometry, these _Openings must be represented as holes within that surface. A hole is represented by an interior ring within the corresponding surface geometry object. According to GML3, the points have to be specified in reverse order (exterior boundaries counter-clockwise and interior boundaries clockwise when looking in opposite direction of the surface’s normal vector). If such an opening is sealed by a Door, a Window, or a ClosureSurface, their outer boundary may consist of the same points as the inner ring (denoting the hole) of the surrounding surface. The embrasure surfaces of an Opening belong to the relevant adjacent _BoundarySurface. If, for example a door seals the Opening, the embrasure surface on the one side of the door belongs to the InteriorWallSurface and on the other side to the WallSurface.

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

**GroundSurfaceType, GroundSurface**

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

**OuterCeilingSurfaceType, OuterCeilingSurface**

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

**WallSurfaceType, WallSurface**

A mostly horizontal surface belonging to the outer bridge shell and having the orientation pointing downwards can be modeled as an *OuterCeilingSurface*.
All parts of the bridge facade belonging to the outer bridge shell can be modelled by the class \textit{WallSurface}.

\textbf{OuterFloorSurfaceType, OuterFloorSurface}

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

\textbf{RoofSurfaceType, RoofSurface}

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

\textbf{ClosureSurfaceType, ClosureSurface}

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

**FloorSurfaceType, FloorSurface**

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

**InteriorWallSurfaceType, InteriorWallSurface**

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

**CeilingSurfaceType, CeilingSurface**

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

**10.5.4 Openings**

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

WindowType, Window

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

DoorType, Door

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

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

BridgeRoomType, BridgeRoom

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

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

BridgeFurnitureType, BridgeFurniture

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

**IntBridgeInstallationType, IntBridgeInstallation**

```xml
<xs:complexType name="IntBridgeInstallationType">
  <xs:complexContent
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="_GenericApplicationPropertyOfIntBridgeInstallation" type="core:_CityObject" minOccurs="0" maxOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

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

10.5.6 Examples

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

Fig. 51: The bridge of Rees, consisting of a Bridge feature and three BridgePart features (left). The bridge contains BridgeConstructionElement and BridgeInstallation features (right).
In the following Fig. 52, the main part of the bridge of Rees is shown as photograph on the left side (source: Harald Halfpapp), and the corresponding part of the LOD2 bridge model is depicted on the the right side (source: District of Recklinghausen / KIT).

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

In CityGML both bridges are modeled as single Bridge object with BridgeConstructionElements and BridgeInstallations. The construction elements of the cable stayed bridge are the footings on both river sides and in the middle of the river, as well as the cables and the pylon. The construction elements of the truss bridge are the footings and the truss itself. Both bridges have several railings which are modeled as BridgeInstallation.

The bridge “Oberbaumbrücke” shown in Fig. 54 is located in the centre of Berlin crossing the river Spree and serves as example for bridges having interior rooms. The real-world bridge is depicted in the left part of Fig. 54, whereas the corresponding CityGML model is shown on the right. The outer geometry of the bridge is modeled as gml:MultiSurface element (lod4MultiSurface property) and is assigned photorealistic textures. Additionally, the interior rooms located in both bridge towers are represented as BridgeRoom objects with solid geometries (gml:Solid assigned through the lod4Solid property). Due to its high geometric accuracy and the representation of the interior structures of both bridge towers, the model is classified as LOD4.
10.5.7 Code lists

The attributes class, function, and usage of the features _AbstractBridge, BridgeConstructionElement, BridgeInstallation, BridgeRoom, BridgeFurniture and IntBridgeInstallation are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.3.

10.5.8 Conformance requirements

Base requirements

1. If a bridge only consists of one (homogeneous) part, it shall be represented by the element Bridge. However, if a bridge is composed of individual structural segments, it shall be modelled as a Bridge element having one or more additional BridgePart elements. Only the geometry and non-spatial properties of the main part of the bridge should be represented within the aggregating Bridge element.

Usage restriction of bridge model components according to different LODs

2. The lodXSolid and lodXMultiSurface, X ∈ [1..4], properties (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType) of _AbstractBridge may be used to geometrically represent the exterior shell of a bridge (as volume or surface model) within each LOD. For LOD1, either lod1Solid or lod1MultiSurface must be used, but not both. Starting from LOD2, both properties may be modelled individually and complementary.

3. Starting from LOD2, the exterior shell of an _AbstractBridge may be semantically decomposed into _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of _AbstractBridge. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of _BoundarySurface are allowed. The boundedBy property (not to be confused with the gml:boundedBy property) shall not be used if the bridge is only represented in LOD1.

If the exterior shell is represented by _BoundarySurface elements, an additional geometric representation as volume or surface model using the lodXSolid and lodXMultiSurface, X ∈ [2..4], properties shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of _BoundarySurface within each LOD using the XLink concept of GML 3.1.1.

4. Starting from LOD2, curve parts of the bridge shell may be represented using the lodXMultiCurve, X ∈ [2..4], property of _AbstractBridge. This property shall not be used if the bridge is only represented in LOD1.

5. Starting from LOD1, the outerBridgeConstruction property (type: BridgeConstructionElementPropertyType) of _AbstractBridge may be used to model BridgeConstructionElement elements. BridgeCon-
structureElement elements shall only be used to represent outer characteristics of a bridge which do not have the significance of bridge parts and are essential from a structural point of view.

6. Starting from LOD2, the geometry of BridgeConstructionElement elements may be semantically classified by _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of BridgeConstructionElement. The boundedBy property (not to be confused with the gml:boundedBy property) shall not be used if the bridge construction element is only represented in LOD1.

7. Starting from LOD2, the outerBridgeInstallation property (type: BridgeInstallationPropertyType) of _AbstractBridge may be used to model BridgeInstallation elements. BridgeInstallation elements shall only be used to represent outer characteristics of a bridge which do not have the significance of bridge parts and are not essential from a structural point of view.

8. Starting from LOD2, the geometry of BridgeInstallation elements may be semantically classified by _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of BridgeInstallation. Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface as subclasses of _BoundarySurface are allowed.

9. Starting from LOD3, openings of _BoundarySurface elements may be modelled using the opening property (type: OpeningPropertyType) of _BoundarySurface. This property shall not be used for _BoundarySurface elements only represented in LOD2. Accordingly, the surface geometry representing a _BoundarySurface in LOD2 must be simply connected.

The opening property of _BoundarySurface may contain or reference _Opening elements. If the geometric location of an _Opening element topologically lies within a surface component of the _BoundarySurface, the opening must also be represented as inner hole of that surface. The embrasure surface of an _Opening element shall belong to the relevant adjacent _BoundarySurface.

10. Starting from LOD4, the interiorBridgeRoom property (type: InteriorBridgeRoomPropertyType) of _AbstractBridge may be used to semantically model the free space inside the bridge by BridgeRoom elements. This property shall not be used if the bridge is only represented in LOD 1 – 3. The BridgeRoom element may be geometrically represented as a surface or volume model, using its lod4Solid or lod4MultiSurface property (gml:SolidPropertyType resp. gml:MultiSurfacePropertyType).

In addition, different parts of the visible surface of a bridge room may be modelled by thematic _BoundarySurface elements. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of _BoundarySurface are allowed. If the visible surface of a room is represented by _BoundarySurface elements, an additional geometric representation as volume or surface model using the lod4Solid and lod4MultiSurface property shall not explicitly define the geometry, but has to reference the according components of the gml:MultiSurface element of _BoundarySurface using the XLink concept of GML 3.1.1.

11. Starting from LOD4, the interiorBridgeInstallation property (type: IntBridgeInstallationPropertyType) of _AbstractBridge may be used to represent immovable objects inside the bridge that are permanently attached to the bridge structure. The interiorBridgeInstallation property shall not be used if the bridge is only represented in LOD 1 – 3. Furthermore, the interiorBridgeInstallation property shall only be used if the object cannot be associated with a BridgeRoom element. In the latter case, the bridgeRoomInstallation property (type: IntBridgeInstallationPropertyType) of the corresponding BridgeRoom element shall be used to represent the object.

12. Starting from LOD4, the geometry of IntBridgeInstallation elements may be semantically classified by _BoundarySurface elements using the boundedBy property (type: BoundarySurfacePropertyType) of IntBridgeInstallation. Only FloorSurface, CeilingSurface, InteriorWallSurface, and ClosureSurface as subclasses of _BoundarySurface are allowed.

Referential integrity

13. The boundedBy property (type: BoundarySurfacePropertyType) of the element _AbstractBridge may contain a _BoundarySurface element inline or an XLink reference to a remote _AbstractBridge element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface ele-
ments are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

Only RoofSurface, WallSurface, GroundSurface, OuterCeilingSurface, OuterFloorSurface and ClosureSurface elements are allowed to be encapsulated or referenced by the boundedBy property of AbstractBridge.

14. The outerBridgeConstruction property (type: BridgeConstructionElementPropertyType) of the element AbstractBridge may contain a BridgeConstructionElement element inline or an XLink reference to a remote BridgeConstructionElement element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the outerBridgeConstruction property may only point to a remote BridgeConstructionElement element (where remote BridgeConstructionElement elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

15. The outerBridgeInstallation property (type: BridgeInstallationPropertyType) of the element AbstractBridge may contain a BridgeInstallation element inline or an XLink reference to a remote BridgeInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the outerBridgeInstallation property may only point to a remote BridgeInstallation element (where remote BridgeInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

16. The interiorBridgeInstallation property (type: IntBridgeInstallationPropertyType) of the element AbstractBridge may contain an IntBridgeInstallation element inline or an XLink reference to a remote IntBridgeInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorBridgeInstallation property may only point to a remote IntBridgeInstallation element (where remote IntBridgeInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

17. The interiorBridgeRoom property (type: InteriorBridgeRoomPropertyType) of the element AbstractBridge may contain a BridgeRoom element inline or an XLink reference to a remote BridgeRoom element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorBridgeRoom property may only point to a remote BridgeRoom element (where remote BridgeRoom elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

18. The consistsOfBridgePart property (type: BridgePartPropertyType) of the element AbstractBridge may contain a BridgePart element inline or an XLink reference to a remote BridgePart element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the consistsOfBridgePart property may only point to a remote BridgePart element (where remote BridgePart elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

19. The address property (type: core:AddressPropertyType) of the element AbstractBridge may contain an core:Address element inline or an XLink reference to a remote core:Address element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the address property may only point to a remote core:Address element (where remote core:Address elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

20. The opening property (type: OpeningPropertyType) of the element BoundarySurface may contain an Opening element inline or an XLink reference to a remote Opening element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the opening property may only point to a remote Opening element (where remote Opening elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

21. The address property (type: core:AddressPropertyType) of the element Door may contain an core:Address element inline or an XLink reference to a remote core:Address element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the address property may only point
to a remote core:Address element (where remote core:Address elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

22. The boundedBy property (type: BoundarySurfacePropertyType) of the element BridgeConstructionElement may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

23. The boundedBy property (type: BoundarySurfacePropertyType) of the element BridgeInstallation may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

24. The boundedBy property (type: BoundarySurfacePropertyType) of the element IntBridgeInstallation may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

25. The boundedBy property (type: BoundarySurfacePropertyType) of the element BridgeRoom may contain a _BoundarySurface element inline or an XLink reference to a remote _BoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _BoundarySurface element (where remote _BoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

26. The interiorFurniture property (type: InteriorFurniturePropertyType) of the element BridgeRoom may contain an BridgeFurniture element inline or an XLink reference to a remote BridgeFurniture element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the interiorFurniture property may only point to a remote BridgeFurniture element (where remote BridgeFurniture elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

27. The bridgeRoomInstallation property (type: IntBridgeInstallationPropertyType) of the element BridgeRoom may contain an IntBridgeInstallation element inline or an XLink reference to a remote IntBridgeInstallation element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the bridgeRoomInstallation property may only point to a remote IntBridgeInstallation element (where remote IntBridgeInstallation elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

28. The lodXImplicitRepresentation, X ∈ [1..4], property (type: core:ImplicitRepresentationPropertyType) of the element BridgeConstructionElement may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, X ∈ [1..4], property may only
point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

29. The lodXImplicitRepresentation, \( X \in [2..4] \), property (type: core:ImplicitRepresentationPropertyType) of the element BridgeInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, \( X \in [2..4] \), property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

30. The lodXImplicitRepresentation, \( X \in [3..4] \), property (type: core:ImplicitRepresentationPropertyType) of the element _Opening may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, \( X \in [3..4] \), property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

31. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element IntBridgeInstallation may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

32. The lod4ImplicitRepresentation property (type: core:ImplicitRepresentationPropertyType) of the element BridgeFurniture may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lod4ImplicitRepresentation property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.6 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 resulted in high-tech bridges of today. The landscapes of many cities are dominated by water, which directly relates to 3D city models. Furthermore, water bodies are important for urban life as subject of recreation and possible hazards as e.g. floods.

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

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

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

![Fig. 55: Illustration of a water body defined in CityGML (graphic: IGG Uni Bonn).](image)

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

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

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

Both LOD0 and LOD1 represent a low level of illustration and high grade of generalisation. Here the rivers are modelled as MultiCurve geometry and brooks are omitted. Seas, oceans and lakes with significant extent are represented as a MultiSurface (Fig. 56). Every WaterBody may be assigned a combination of geometries of different types. Linear water bodies are represented as a network of 3D curves. Each curve is composed of straight line segments, where the line orientation denotes the flow direction (water flows from the first point of a curve, e.g. a gml:LineString, to the last). Areal objects like lakes or seas are represented by 3D surface 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 gml:Solid in LOD2 or higher, the surface geometries of the corresponding thematic WaterClosureSurface, WaterGroundSurface, and WaterSurface objects must coincide with the exterior shell of the gml:Solid. This can be ensured, if for each LOD X the respective lodXSolid representation (where X is between 2 and 4) does not redundantly define the geometry, but instead references the corresponding polygons (using GML3’s XLink mechanism) of the lodXSurface elements (where X is between 2 and 4) of WaterClosureSurface, WaterGroundSurface, and WaterSurface.

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

Every object of the class WaterSurface, WaterClosureSurface, and WaterGroundSurface must have at least one associated surface geometry. This means, that every WaterSurface, WaterClosureSurface, and 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 6.5). The rings defining the WaterSurface polygons implicitly delineate the intersection of the water body with the terrain or basin.

**XML namespace**

The XML namespace of the CityGML WaterBody module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/waterbody/2.0. Within the XML Schema definition of the WaterBody module, this URI is also used to identify the default namespace.
10.6.1 Water body

AbstractWaterObjectType, _WaterObject

WaterBodyType, WaterBody

10.6.2 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. 55).

1. Boundary class “Air to Water”. The WaterSurface is mandatory to the model and usually is registered using photogrammetric analysis or mapping exploration. The representation may vary due to tidal flats or changing water levels, which can be reflected by including different static water surfaces having different waterLevels (gml:CodeType), as for example highest flooding event, mean sea level, or minimum water level. This offers the opportunity to describe significant water surfaces due to levels that are important for certain representations e.g. in tidal zones.

2. Boundary class “Water to Ground”. The WaterGroundSurface may be known by sonar exploration or other depth measurements. Also part of the ground surface is the boundary “Water to Construction”. The ground surface might be identical to the underwater terrain model, but also describes the contour to other underwater objects. The usefulness of this concept arises from the existence of water defence constructions like sluices, sills, flood barrage or tidal power stations. The use of WaterGroundSurface as boundary layer to man-made constructions is relevant in urban situations, where such objects may enclose the modeled water body completely, for example fountains and swimming pools. The WaterSurface objects together with the WaterGroundSurface objects enclose the WaterBody as a volume.
3. Boundary class “Water to Water”. The WaterClosureSurface is an optional feature that comes in use when the union of the WaterSurfaces and WaterGroundSurfaces of a water body does not define a closed volume. The WaterClosureSurface is then used to complete the enclosure of water volumes and to separate water volumes from those where only the surface is known. This might occur, where the cross section and ground surface of rivers is partly available during its course.

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

AbstractWaterBoundarySurfaceType, _WaterBoundarySurface

```
<xs:complexType name="AbstractWaterBoundarySurfaceType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="lod2Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod3Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element name="lod4Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfWaterBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

WaterSurfaceType, WaterSurface

```
<xs:complexType name="WaterSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element name="waterLevel" type="gml:CodeType" minOccurs="0"/>
        <xs:element ref="_GenericApplicationPropertyOfWaterSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

WaterGroundSurfaceType, WaterGroundSurface

```
<xs:complexType name="WaterGroundSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

WaterClosureSurfaceType, WaterClosureSurface

```
<xs:complexType name="WaterClosureSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
10.6.3 Code lists

The attributes class, function, and usage of the feature WaterBody as well as the attribute waterLevel of the feature WaterSurface are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.9.

10.6.4 Conformance requirements

Base requirements

1. For LOD0 and LOD1, the geometry of a WaterBody may be modelled as a linear network using gml:MultiCurve geometry elements. In that case, each gml:MultiCurve shall be composed of straight line segments, where the line orientation denotes the flow direction. The flow direction is from the first point of a line segment to its last point.

2. Starting from LOD2, the exterior shell of a WaterBody may be semantically decomposed into _WaterBoundarySurface elements using the boundedBy property (type: BoundedByWaterSurfacePropertyType) of WaterBody. The boundedBy property shall not be used if the water body is only represented in lower LODs. If the exterior shell is represented by _WaterBoundarySurface elements, an additional geometric representation as volume model using the lodXSolid, $X \in [2..4]$, property of WaterBody shall not explicitly define the geometry, but has to reference the according gml:_Surface elements of the _WaterBoundarySurface objects within each LOD using the XLink concept of GML 3.1.1.

3. Each _WaterBoundarySurface element must have at least one associated surface geometry given by the lodXSurface, $X \in [2..4]$, properties of _WaterBoundarySurface.

4. _WaterBoundarySurface elements shall only be included as parts of corresponding WaterBody elements. They may not be given as stand-alone city objects within a CityGML model.

Referential integrity

5. The boundedBy property (type: BoundedByWaterSurfacePropertyType) of the element WaterBody may contain a _WaterBoundarySurface element inline or an XLink reference to a remote _WaterBoundarySurface element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the boundedBy property may only point to a remote _WaterBoundarySurface element (where remote _WaterBoundarySurface elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.7 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 for the application of geometric route planning algorithms. This can be useful to determine restrictions and manoeuvres required along a transportation route. This information can also be employed for trajectory planning of mobile robots in the real world or the automatic placement of avatars (virtual people) or vehicle models in 3D visualisations and training simulators. The transportation model of CityGML is provided by the thematic extension module Transportation (cf. chapter 7).

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

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

Fig. 59 depicts the UML diagram of the transportation model, for the XML schema definition see annex A.10.
The road itself is represented as a `TransportationComplex`, which is further subdivided into `TrafficAreas` and `AuxiliaryTrafficAreas`. The `TrafficAreas` are those elements, which are important in terms of traffic usage, like car driving lanes, pedestrian zones and cycle lanes. The `AuxiliaryTrafficAreas` are describing further elements of the road, like kerbstones, middle lanes, and green areas.

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

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

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

<table>
<thead>
<tr>
<th>Example:</th>
<th>Country road</th>
<th>Motorway entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransportationComplex – Function</td>
<td>road</td>
<td>road</td>
</tr>
<tr>
<td>TrafficArea – Usage</td>
<td>car, truck, bus, taxi, motorcycle</td>
<td>car, truck, bus, taxi, motorcycle</td>
</tr>
<tr>
<td>TrafficArea – Function</td>
<td>driving lane</td>
<td>motorway_entry</td>
</tr>
<tr>
<td>TrafficArea – SurfaceMaterial</td>
<td>asphalt</td>
<td>concrete</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example:</th>
<th>Runway of an airport</th>
<th>Apron of an airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransportationComplex – Function</td>
<td>road</td>
<td>apron</td>
</tr>
<tr>
<td>TrafficArea – Usage</td>
<td>aeroplane</td>
<td>aeroplane, car, truck, bus, pedestrian</td>
</tr>
<tr>
<td>TrafficArea – Function</td>
<td>airport – runway</td>
<td>airport – apron</td>
</tr>
<tr>
<td>TrafficArea – SurfaceMaterial</td>
<td>concrete</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. The street name is also stored within the gml:name property of the Road feature.

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

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

![Diagram of TransportationComplex in LOD0, 1, and 2-4](source: Rheinmetall Defence Electronics)

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

**XML namespace**

The XML namespace of the CityGML Transportation module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/transportation/2.0. Within the XML Schema definition of the Transportation module, this URI is also used to identify the default namespace.

10.7.1 Transportation complex

AbstractTransportationObjectType, _TransportationObject

```xml
<xs:complexType name="AbstractTransportationObjectType" abstract="true">
  <xs:complexContent>
    ...
  </xs:complexContent>
</xs:complexType>
```
TransportationComplexType, TransportationComplex

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

10.7.2 Subclasses of transportation complexes

TrackType, Track

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

```xml
c<x:complexType name="RoadType">
  <x:complexContent>
    <x:extension base="TransportationComplexType">
      <x:sequence>
        <x:element ref="_GenericApplicationPropertyOfRoad" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x:complexType>
```

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

```xml
c<x:complexType name="RailwayType">
  <x:complexContent>
    <x:extension base="TransportationComplexType">
      <x:sequence>
        <x:element ref="_GenericApplicationPropertyOfRailway" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x:complexType>
```

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

SquareType, Square

```xml
c<x:complexType name="SquareType">
  <x:complexContent>
    <x:extension base="TransportationComplexType">
      <x:sequence>
        <x:element ref="_GenericApplicationPropertyOfSquare" minOccurs="0" maxOccurs="unbounded"/>
      </x:sequence>
    </x:extension>
  </x:complexContent>
</x: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.
10.7.3 Subdivisions of transportation complexes

TrafficAreaType, TrafficArea

```xml
<xs:complexType name="TrafficAreaType">
    <xs:complexContent base="AbstractTransportationObjectType">
        <xs:extension base="TrafficAreaType">
            <xs:sequence>
                <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="_GenericApplicationPropertyOfTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

AuxiliaryTrafficAreaType, AuxiliaryTrafficArea

```xml
<xs:complexType name="AuxiliaryTrafficAreaType">
    <xs:complexContent base="AbstractTransportationObjectType">
        <xs:extension base="AuxiliaryTrafficAreaType">
            <xs:sequence>
                <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="surfaceMaterial" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="_GenericApplicationPropertyOfAuxiliaryTrafficArea" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

10.7.4 Code lists

The attributes class, function, and usage of the features TransportationComplex, TrafficArea and AuxiliaryTrafficArea as well as the attribute surfaceMaterial of the features TrafficArea and AuxiliaryTrafficArea are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.8.

10.7.5 Conformance requirements

Base requirements

1. For LOD0, the geometry of a TransportationComplex shall be modelled using GML line objects representing the centerline of the transportation complex. The line objects shall establish a linear network. Thus, the lod0Network property (type: gml:GeometricComplexPropertyType) of the element TransportationComplex may only contain or reference an appropriate curve geometry element.

2. Starting from LOD2, the trafficArea property (type: TrafficAreaPropertyType) as well as the auxiliaryTrafficArea property (type: AuxiliaryTrafficAreaPropertyType) of the element TransportationComplex...
plex may be used. These properties shall not be used if the transportation complex is only represented in lower LODs.

**Referential integrity**

3. The `trafficArea` property (type: `TrafficAreaPropertyType`) of the element `TransportationComplex` may contain a `TrafficArea` element inline or an XLink reference to a remote `TrafficArea` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `trafficArea` property may only point to a remote `TrafficArea` element (where remote `TrafficArea` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

4. The `auxiliaryTrafficArea` property (type: `TrafficAreaPropertyType`) of the element `TransportationComplex` may contain an `AuxiliaryTrafficArea` element inline or an XLink reference to a remote `AuxiliaryTrafficArea` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `auxiliaryTrafficArea` property may only point to a remote `AuxiliaryTrafficArea` element (where remote `AuxiliaryTrafficArea` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.8 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, for example forest fire, urban aeration or micro climate. The model could be used, for example 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 is defined by the thematic extension module Vegetation (cf. chapter 7).

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. 63). Single vegetation objects are modelled by the class SolitaryVegetationObject, whereas 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 extent of the vegetation. For example regarding forests, a MultiSolid representation might be more appropriate. The UML diagram of the vegetation model is depicted in Fig. 64, for the XML schema definition see below and annex A.12.

A SolitaryVegetationObject may have the attributes class, function, usage, species height, trunkDiameter and crownDiameter. The attribute class contains the classification of the object or plant habit, e.g. tree, bush, grass, and can occur only once (see chapter 10.8.4 and annex C.7). The attribute species defines the species’ name, for example “Abies alba”, and can occur at most once (see chapter 10.8.4 and annex C.7). The optional attributes function and usage denotes the intended respectively real purpose of the object, for example botanical monument, and can occur multiple times. The possible attribute values for class, species, function, and usage can be provided in a code list. The attribute height contains the relative height of the object. The attributes crownDiameter and trunkDiameter represent the plant crown and trunk diameter respectively. 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, usage and averageHeight. The plant community of a PlantCover is represented by the attribute class. The values of this attribute can be specified in a code list (cf. chapter 10.8.4 and annex C.7) whose values should not only describe one plant type or species, but denote 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 attributes function and usage indicate the intended respectively real purpose of the object, for example national forest, and can occur multiple times. The attribute averageHeight denotes the average relative vegetation height.
Since both SolitaryVegetationObject and PlantCover are derived from _CityObject, they inherit 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 6.7).

Fig. 64: UML diagram of vegetation objects in CityGML. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Vegetation module.

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 (cf. chapter 8.2). Solitary vegetation objects probably are one of the most important features where implicit geometries are appropriate, since the shape of most types of vegetation objects, such as trees of the same species, can be treated as identical in most cases. Furthermore, season dependent appearances may be mapped using ImplicitGeometry. For visualisation purposes, only the content of the library object defining the object’s shape and appearance has to be swapped (cf. Fig. 65).

A SolitaryVegetationObject or a PlantCover may have a different geometry in each LOD. Whereas a SolitaryVegetationObject is associated with the gml:_Geometry class representing an arbitrary GML geometry (by the relation lodXGeometry, X ∈ [1..4]), a PlantCover is restricted to be either a gml:MultiSolid or a gml:MultiSurface. An example of a PlantCover modelled as gml:MultiSolid is a ‘solid forest model’, see Fig. 66.
Fig. 66: Example for the visualisation/modelling of a solid forest (source: District of Recklinghausen).

**XML namespace**

The XML namespace of the CityGML *Vegetation* module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/vegetation/2.0. Within the XML Schema definition of the *Vegetation* module, this URI is also used to identify the default namespace.

### 10.8.1 Vegetation object

**AbstractVegetationObjectType, _VegetationObject**

```xml
<xs:complexType name="AbstractVegetationObjectType" abstract="true">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfVegetationObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

### 10.8.2 Solitary vegetation objects

**SolitaryVegetationObjectType, SolitaryVegetationObject**

```xml
<xs:complexType name="SolitaryVegetationObjectType">
  <xs:extension base="AbstractVegetationObjectType">
    <xs:sequence>
      <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
      <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="species" type="gml:CodeType" 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:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:extension>
</xs:complexType>
```
10.8.3 Plant cover objects

PlantCoverType, PlantCover

10.8.4 Code lists

The attributes class, function, and usage of the features PlantCover and SolitaryVegetationObject as well as the attribute species of the feature SolitaryVegetationObject are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.7.

10.8.5 Example 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. The attribute values of the class and species attributes are taken from code lists proposed by the SIG 3D which are presented in annex C.7.

<SolitaryVegetationObject>
  <class codeSpace="http://www.sig3d.org/codelist/standard/vegetation/2.0/SolitaryVegetationObject_class.xml">1070</class>
  <species codeSpace="http://www.sig3d.org/codelist/standard/vegetation/2.0/SolitaryVegetationObject_species.xml">1040</species>
  <height uom="m">8</height>
  <trunkDiameter uom="m">0.7</trunkDiameter>
  <crownDiameter uom="m">8</crownDiameter>
  <lod1ImplicitRepresentation>
    <core:ImplicitGeometry>
      <core:mimeType>1010</core:mimeType>
      <core:libraryObject>urn:sig3d:tree.wrl</core:libraryObject>
      <core:referencePoint>
      </core:referencePoint>
    </core:ImplicitGeometry>
  </lod1ImplicitRepresentation>
</SolitaryVegetationObject>
10.8.6 Conformance requirements

Referential integrity

- The \textit{lodXImplicitRepresentation}, X \in [1..4], property (type: \textit{core:ImplicitRepresentationPropertyType}) of the element \textit{SolitaryVegetationObject} may contain a \textit{core:ImplicitGeometry} element inline or an XLink reference to a remote \textit{core:ImplicitGeometry} element using the XLink concept of GML 3.1.1. In the latter case, the \textit{xlink:href} attribute of the \textit{lodXImplicitRepresentation}, X \in [1..4], property may only point to a remote \textit{core:ImplicitGeometry} element (where remote \textit{core:ImplicitGeometry} elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.9 City furniture

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, flower buckets, advertising columns, benches, delimitation stakes, or bus stops (Fig. 67, Fig. 68). 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 of, for example 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. The city furniture model of CityGML is defined by the thematic extension module CityFurniture (cf. chapter 7).

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

The UML diagram of the city furniture model is depicted in Fig. 69, for the XML schema definition see below and annex A.5.

The class CityFurniture may have the attributes class, function and usage. Their possible values can be specified in corresponding code lists (chapter 10.9.2 and annex C.4). 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 to (e.g. transportation, traffic regulation, architecture), and can occur multiple times. The attribute usage denotes the real purpose of the object.
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 (cf. chapter 6.7) and may be augmented by generic attributes using CityGML’s Generics module (cf. chapter 10.12). For ExternalReferences city furniture objects can have links to external thematic databases. Thereby, semantic information of the objects, which cannot be modelled in CityGML, can be transmitted and used in the 3D city model for further processing, for example 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 their specific geometry, but in most cases the same kind of object has an identical geometry. The geometry of 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). Following the concept of ImplicitGeometry the geometry of a prototype city furniture is stored only once in a local coordinate system and referenced by other city furniture features (see chapter 8.2). Spatial information of city furniture objects can be taken, for example, from city maps or from public and private external information systems.

In order to specify the exact intersection of the DTM with the 3D geometry of a city furniture object, the latter can have a TerrainIntersectionCurve (TIC) for each LOD (cf. chapter 6.5). This allows for ensuring a smooth transition between the DTM and the city furniture object.

**XML namespace**

The XML namespace of the CityGML CityFurniture module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityfurniture/2.0. Within the XML Schema definition of the CityFurniture module, this URI is also used to identify the default namespace.

**10.9.1 City furniture object**

CityFurnitureType, CityFurniture
10.9.2 Code lists

The attributes class, function, and usage of the feature CityFurniture are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.4.

10.9.3 Example CityGML dataset

The following example of a CityGML dataset is an extract of the model of a delimitation stake in LOD3 and contains the attributes class = 1000 and function = 1520 (delimitation stake) whose coded attribute values are taken from a code list proposed by the SIG 3D (cf. annex C.4). The delimitation stake with the object ID stake0815 has an ExternalReference pointing to a cadastre object within the German ALKIS database (www.adv-online.de) which is identified by the URI urn:adv:oid:DEHE123400007001.

The example dataset shows the geometry of the top surface (gml:id “cover”) and of the left surface (gml:id “surfLeft”) of the stake which are both depicted in Fig. 70. The top surface is assigned a constant material (white color) and the left surface is textured using the texture image stake.gif by denoting the relevant texture coordinates. Both surface appearances are modelled using the CityGML Appearance module (cf. chapter 9).

Fig. 70: Example of a simple city furniture object (source: District of Recklinghausen).
10.9.4 Conformance requirements

Referential integrity

- The lodXImplicitRepresentation, X \in [1..4], property (type: core:ImplicitRepresentationPropertyType) of the element CityFurniture may contain a core:ImplicitGeometry element inline or an XLink reference to a remote core:ImplicitGeometry element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the lodXImplicitRepresentation, X \in [1..4], property may only point to a remote core:ImplicitGeometry element (where remote core:ImplicitGeometry elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.10 Land use

LandUse objects can be used to describe areas of the earth’s surface dedicated to a specific land use, but also to describe areas of the earth’s surface having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, etc (i.e. the physical appearance). Land use and land cover are different concepts; the first describes human activities on the earth’s surface, the second describes its physical and biological cover. However, the two are interlinked and often mixed in practice. LandUse objects in CityGML represent both concepts: They can be employed to represent, parcels, spatial planning objects, recreational objects and objects describing the physical characteristics of an area, in 3D (e.g. wetlands). Fig. 71 shows the UML diagram of land use objects, for the XML schema definition see chapter 10.10.1 and annex A.8. The land use model of CityGML is provided by the thematic extension module LandUse (cf. chapter 7).

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 can be specified in a code list (cf. annex C.5). The attribute function defines the purpose of the object or their nature, like e.g. cornfield or heath, 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 appearance properties like textures or colors (using CityGML’s appearance model, cf. chapter 9).

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. 72 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.
XML namespace

The XML namespace of the CityGML LandUse module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/landuse/2.0. Within the XML Schema definition of the LandUse module, this URI is also used to identify the default namespace.

10.10.1 Land use object

LandUseType, LandUse

```xml
<xs:complexType name="LandUseType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" 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:element ref="_GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

10.10.2 Code lists

The attributes class, function, and usage of the feature LandUse are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.5.
10.10.3 Conformance requirements

Base requirements

- The `gml:MultiSurface` geometry element representing the geometry of a `LandUse` object must be given with 3D coordinate values within each LOD.
City object groups

The CityGML grouping concept has been introduced in chapter 6.8. CityObjectGroups are modelled using the 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. 73 shows the UML diagram for the class CityObjectGroup, for the XML schema see annex A.6. The grouping concept of CityGML is defined by the thematic extension module CityObjectGroup (cf. chapter 7).

The class CityObjectGroup has the optional attributes class, function and usage. 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. 9 in chapter 8.1. This may be used to represent a generalised geometry generated from the members geometries.

The parent association linking a CityObjectGroup to a _CityObject allows for the modelling of a generic hierarchical grouping concept. Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The parent association links to the aggregate, while the parts are given by the group members. This concept is used, for example, to represent storeys in buildings (see section 10.3.6: Modelling building storeys using CityObjectGroups).

XML namespace

The XML namespace of the CityGML CityObjectGroup module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/cityobjectgroup/2.0. Within the XML Schema definition of the CityObjectGroup module, this URI is also used to identify the default namespace.

10.11.1 City object group

CityObjectGroupType, CityObjectGroup

```xml
<xs:complexType name="CityObjectGroupType">  
  <xs:complexContent>  
    <xs:extension base="core:AbstractCityObjectType">  
      <xs:sequence>  
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>  
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>  
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>  
        <xs:element name="groupMember" type="CityObjectGroupMemberType" minOccurs="0" maxOccurs="unbounded"/>  
      </xs:sequence>  
    </xs:extension>  
  </xs:complexContent>  
</xs:complexType>
```
10.11.2 Code lists

The attributes class, function, and usage of the feature CityObjectGroup are specified as gml:CodeType. The values of these properties can be enumerated in code lists. Proposals for corresponding code lists can be found in annex C.10.

10.11.3 Conformance requirements

Base requirements

1. No cyclic groupings shall be included within a CityGML instance document.

Referential integrity

2. The groupMember property (type: CityObjectGroupMemberType) of the element CityObjectGroup may contain a core:_CityObject element inline or an XLink reference to a remote core:_CityObject element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the groupMember property may only point to a remote core:_CityObject element (where remote core:_CityObject elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.

3. The parent property (type: CityObjectGroupParentType) of the element CityObjectGroup may contain a core:_CityObject element inline or an XLink reference to a remote core:_CityObject element using the XLink concept of GML 3.1.1. In the latter case, the xlink:href attribute of the parent property may only point to a remote core:_CityObject element (where remote core:_CityObject elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.12 Generic city objects and attributes

The concept of generic city objects and attributes allows for the storage and exchange of 3D objects which are not covered by any explicitly modelled thematic class within CityGML or which require attributes not represented in CityGML. These generic extensions to the CityGML data model are realised by the classes GenericCityObject and _genericAttribute defined within the thematic extension module Generics (cf. chapter 7). In order to avoid problems concerning semantic interoperability, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.

Fig. 74 shows the UML diagram of generic objects and attributes. For XML schema definition see below and annex A.7.

Generic attributes are name-value pairs associated with a city object. Each generic attribute has a mandatory name identifier which can be freely chosen. The data type of the attribute value may be String, Integer, Double (floating point number), URI, Date, and gml:MeasureType. The attribute type is defined by the selection of the particular subclass of _genericAttribute, for example stringAttribute, intAttribute, etc. A measureAttribute facilitates the representation of measured values. Its value is of the structured type gml:MeasureType which provides an optional attribute uom (units of measure) of type xs:anyURI that points to a reference system for the amount.

Generic attributes can be grouped under a common name using a genericAttributeSet. The genericAttributeSet class is derived from _genericAttribute and thus is also realized as generic attribute. Its value is the set of contained generic attributes and its name property provides a name identifier for the entire set. Since genericAttributeSet is itself a generic attribute, it may also be contained in a generic attribute set facilitating a recursive nesting of arbitrary depth. The optional codeSpace attribute (of type xs:anyURI) of genericAttributeSet is used to associate the attribute set with an authority, e.g. the organisation or community who defined the attribute set and its contained attributes. By this means, generic attribute sets can be clearly distinguished even if they share the same name.

In order to model generic attributes, the abstract base class _CityObject defined within the CityGML Core module is augmented by the additional property element _genericAttribute using CityGML’s Application Domain Extension mechanism (cf. chapter 6.12). By this means, each thematic subclass of _CityObject inherits this property and, thus, may be assigned an arbitrary number of generic attributes in order to represent additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model.
Thus, the Generics module has a deliberate impact on all CityGML extension modules defining thematic subclasses of GenericCityObject.

A GenericCityObject may have the attributes class, function and usage defined as gml:CodeType. The class attribute allows an object classification within the thematic area such as 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.

The geometry of a GenericCityObject can either be an explicit GML3 geometry or an ImplicitGeometry (see chapter 8.2). In the case of an explicit geometry the object can have only one geometry for each LOD, which may be an arbitrary 3D geometry object (class gml:Geometry, which is the base class of all GML geometries, lodXGeometry, X ∈ [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. Alternatively the shape can be specified as a 3D GML3 geometry with local cartesian coordinates using the property relativeGeometry (further details are given in chapter 8.2).

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 6.5). This is important for 3D visualisation but also for certain applications like driving simulators. For example, if a city wall (e.g., the Great Wall of China) should be represented as a GenericCityObject, a smooth transition between the DTM and the city wall would have to be ensured.

XML namespace

The XML namespace of the CityGML Generics module is identified by the Uniform Resource Identifier (URI) http://www.opengis.net/citygml/generics/2.0. Within the XML Schema definition of the Generics module, this URI is also used to identify the default namespace.

10.12.1 Generic city object

GenericCityObjectType, GenericCityObject

```xml
<xs:complexType name="GenericCityObjectType">
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0Geometry" type="gml:GeometryPropertyType" 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:element name="lod0TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
        <xs:element name="lod0ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
10.12.2 Generic attributes

AbstractGenericAttributeType, _genericAttribute, StringAttributeType, stringAttribute, etc.

```xml
<xs:complexType name="AbstractGenericAttributeType" abstract="true">
  <xs:sequence/>
  <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>

<xs:element name="_genericAttribute" type="AbstractGenericAttributeType" abstract="true" substitutionGroup="core:_GenericApplicationPropertyOfCityObject"/>

<xs:complexType name="StringAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <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:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <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:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <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="AbstractGenericAttributeType">
      <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="AbstractGenericAttributeType">
      <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"/>

<xs:complexType name="MeasureAttributeType">
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element name="value" type="xs:double"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="measureAttribute" type="MeasureAttributeType" substitutionGroup="_genericAttribute"/>
```

```
10.12.3 Code lists

The attributes class, function, and usage of the feature `GenericCityObject` are specified as `gml:CodeType`. The values of these properties can be enumerated in code lists.

10.12.4 Conformance requirements

Usage restriction for generic city objects and attributes

1. The `GenericCityObject` element may only be used to model thematic city objects which are not provided by any other CityGML module and, thus, are not covered by the overall CityGML data model. If an appropriate thematic class is available though, this thematic class shall be used instead, and, consequently, the corresponding CityGML module has to be employed by the CityGML instance document.

2. The `_genericAttribute` property of the element `core:CityObject` shall only be used to describe additional properties of features not represented by the explicitly modelled thematic classes of the CityGML data model. Thus, generic attributes shall only be modelled if the appropriate thematic class representing the feature does not offer a feasible property.

Referential integrity

3. The `lodXImplicitRepresentation`, $X \in [0..4]$, property (type: `core:ImplicitRepresentationPropertyType`) of the element `GenericCityObject` may contain a `core:ImplicitGeometry` element inline or an XLink reference to a remote `core:ImplicitGeometry` element using the XLink concept of GML 3.1.1. In the latter case, the `xlink:href` attribute of the `lodXImplicitRepresentation`, $X \in [0..4]$, property may only point to a remote `core:ImplicitGeometry` element (where remote `core:ImplicitGeometry` elements are located in another document or elsewhere in the same document). Either the contained element or the reference must be given, but neither both nor none.
10.13 Application Domain Extensions (ADE)

CityGML has been designed as an application independent information model and exchange format for 3D city and landscape models. However, specific applications typically have additional information needs to be modelled and exchanged. In general, there are two different approaches to combine city model data and application data:

1. Embed the CityGML objects into a (larger) application framework and establish the connection between application data and CityGML data within the application framework. For example, CityGML data fragments may be embedded into the application’s XML data files or stored as attributes of application objects according to the application’s data model.

2. Incorporate application specific information into the CityGML instance documents. This approach is especially feasible, if the application specific information follows essentially the same structure as defined by the CityGML schema. This is the case, if the application data could be represented by additional attributes of CityGML objects and only some new feature types would have to be defined.

In the following, we will focus on the second option, as only this approach lies within the scope of this specification. Generic attributes and objects have already been introduced as a first possibility to support the exchange of application specific data (see section 10.12). Whereas they allow to extend CityGML without changing its XML schema definition, this flexibility has some disadvantages:

- Generic attributes and objects may occur arbitrarily in the CityGML instance documents, but there is no formal specification of the names, datatypes, and multiplicities. Thus, there is no guarantee for an application that a specific instance of a generic attribute is included a minimum or maximum number of times per CityGML feature. Unlike the predefined CityGML objects, the concrete layout and occurrence of generic objects and attributes cannot be validated by an XML parser. This may reduce semantic interoperability.

- Naming conflicts of generic attributes or objects can occur, if the CityGML instance documents should be augmented by specific information from different applications simultaneously.

- There is only a limited number of predefined data types that can be used for generic attributes. Also the structure of generic objects might not be appropriate to represent more complex objects.

If application specific information are well-structured, it is desirable to represent them in a systematic way, i.e. by the definition of an extra formal schema based on the CityGML schema definitions. Such an XML schema is called a CityGML Application Domain Extension (ADE). It allows to validate instance documents both against the extended CityGML and the ADE schema and therefore helps to maintain semantic and syntactic interoperability between different systems working in the same application field. In order to prevent naming conflicts, every ADE has to be defined within its own namespace which must differ from the namespaces associated with the CityGML modules. An ADE schema may extend one or more CityGML module schemas. The relevant CityGML module schemas have to be imported by the ADE schema.

The ADE concept defines a special way of extending existing CityGML feature types which allows to use different ADEs within the same instance document simultaneously (see below). For example, the specification of ADEs can be useful in the following application fields: cultural heritage (extension of abstract class _CityObject e.g. by time period information and monument protection status); representation of subsurface objects (tunnel, underpass) or city lighting (light sources like street lamps and house lights); real estate management (economic parameters of the CityGML features; inclusion of attributes defined for real estate assets as defined by OSCRE); utility networks (as topographic features); additional building properties as defined by the U.S. national building information model standard (NBIMS).

10.13.1 Technical principle of ADEs

Each ADE is specified by its own XML schema file. The target namespace is provided by the information community who specifies the CityGML ADE. This is typically not the OGC or the SIG 3D. The namespace should be in the control of this information community and must be given as a previously unused and globally
unique URI. This URI will be used in CityGML ADE instance documents to distinguish extensions from CityGML base elements. As the URI refers to the information community it also denotes the originator of the employed ADE.

The ADE’s XML schema file must be available (or accessible on the Internet) to everybody creating and parsing CityGML instance documents including these ADE specific augmentations.

An ADE XML schema can define various extensions to CityGML. However, all extensions shall belong to one of the two following categories:

1. New feature types are defined within the ADE namespace and are based on CityGML abstract or concrete classes. In general, this mechanism follows the same principles as the definition of application schemas for GML. This means, that new feature types have to be derived from existing (here: CityGML) feature types. For example, new feature types could be defined by classes derived from the abstract classes like _CityObject or _AbstractBuilding or the concrete class CityFurniture. The new feature types then automatically inherit all properties (i.e. attributes) and associations (i.e. relations) from the respective CityGML superclasses.

2. Existing CityGML feature types are extended by application specific properties (in the ADE namespace). These properties may have simple or complex data types. Also geometries or embedded features (feature properties) are possible. The latter can also be used to model relations to other features.

In this case, extension of the CityGML feature type is not being realised by the inheritance mechanism of XML schema. Instead, every CityGML feature type provides a “hook” in its XML schema definition, that allows to attach additional properties to it by ADEs. This “hook” is implemented as a GML property of the form “_GenericApplicationPropertyOf<Featuretypename>” where <Featuretypename> is equal to the name of the feature type definition in which it is included. The datatype for these kinds of properties is always “xsd:anyType” from the XSD namespace. The minimum occurrence of the “_GenericApplicationPropertyOf<Featuretypename>” is 0 and the maximum occurrence unbounded. This means, that the CityGML schema allows that every CityGML feature may have an arbitrary number of additional properties with arbitrary XML content with the name “_GenericApplicationPropertyOf<Featuretypename>”. For example, the last property in the definition of the CityGML feature type LandUse is the element _GenericApplicationPropertyOfLandUse (cf. chapter 10.10.1).

Such properties are called “hooks” to attach application specific properties, because they are used as the head of a substitution group by ADEs. Whenever an ADE wants to add an extra property to an existing CityGML feature type, it should declare the respective element with the appropriate datatype within the ADE namespace. In the element declaration this element shall be explicitly assigned to the substitution group defined by the corresponding “_GenericApplicationPropertyOf<Featuretypename>” in the corresponding CityGML module namespace. An example is given in the following subsection.

By following this concept, it is possible to specify different ADEs for different information communities. Every ADE may add its specific properties to the same CityGML feature type as they all can belong to the same substitution group. This allows to have CityGML instance documents where CityGML features contain additional information from different ADEs simultaneously.

Please note that usage of ADEs introduces an extra level of complexity as data files may contain mixed information (features, properties) from different namespaces, not only from the GML and CityGML module namespaces. However, extended CityGML instance documents are quite easy to handle by applications that are not “schema-aware”, i.e. applications that do not parse and interpret GML application schemas in a generic way. These applications can simply skip anything from a CityGML instance document that is not from a CityGML module or GML namespace. Thus, a building is still represented by the <bldg:Building> element with the standard CityGML properties, but with possibly some extra properties from different namespaces. Also features from a different namespace than those declared by CityGML modules or GML could be skipped (e.g. by a viewer application).

10.13.2 Example ADE

In this section, the ADE mechanism is illustrated by a short example, which deals with the application of virtual 3D city models to generate noise pollution maps. In our example, two extensions of CityGML are required for
this task: buildings have to be extended to represent a “noise reflection correction” value and the number of inhabitants. As a new feature type noise barriers have to be defined which also have a “noise reflection correction” value.

The XSD schema which has to be defined to implement this model declares a new namespace for the noise extension (http://www.citygml.org/ade/noise_de/2.0). Additionally, the namespaces of the extended CityGML modules are declared (for corresponding prefixes see chapter 4.3 and chapter 7), and the respective schema definition files are imported. The XML schema adds the elements buildingReflectionCorrection and buildingHabitants, both being members of the substitution group bldg:_GenericApplicationPropertyOfAbstractBuilding which is defined by the CityGML Building module. Thus, both elements may be used as child elements of CityGML building features. Noise barriers are represented as NoiseCityFurnitureSegment elements. The corresponding type NoiseCityFurnitureSegmentType is defined as subtype of the CityGML abstract type core:AbstractCityObjectType provided by the CityGML Core module, applying the usual subtyping mechanism of XML and XSD. A further element noiseCityFurnitureSegmentProperty is added as a member of the substitution group frn:_GenericApplicationPropertyOfCityFurniture. By this means, noise barriers may be modelled as child elements of CityGML city furniture objects.

The XSD file for this example CityGML Noise ADE is given by the following excerpt (the complete CityGML Noise ADE is given in Annex H):

```xml
<xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:frn="http://www.opengis.net/frn"
    xmlns="http://www.citygml.org/ade/noise_de/2.0"
    xmlns:core="http://www.opengis.net/citygml/2.0"
    xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
    xmlns:frn:bldg="http://www.opengis.net/citygml/building/2.0"
    xmlns:frn:core="http://www.opengis.net/citygml/core/2.0"
    xs:targetNamespace="http://www.citygml.org/ade/noise_de/2.0"
    elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:import namespace="http://www.opengis.net/citygml/building/2.0" schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityfurniture/2.0" schemaLocation="http://schemas.opengis.net/citygml/cityfurniture/2.0/cityFurniture.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/2.0" schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/2.0" schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/2.0" schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/2.0" schemaLocation="http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"/>
  ...

  <xs:element name="buildingReflection" type="xs:string"
      substitutionGroups="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xs:element name="buildingHabitants" type="xs:positiveInteger"
      substitutionGroups="bldg:_GenericApplicationPropertyOfAbstractBuilding"/>
  <xs:element name="noiseCityFurnitureSegmentProperty" type="NoiseCityFurnitureSegmentPropertyType"
      substitutionGroups="frn:_GenericApplicationPropertyOfCityFurniture"/>

  <xs:complexType name="NoiseCityFurnitureSegmentPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="NoiseCityFurnitureSegment" minOccurs="0"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>

  <xs:complexType name="NoiseCityFurnitureSegmentType">
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="reflection" type="xs:string" minOccurs="0"/>
          <xs:element name="reflectionCorrection" type="gml:MeasureType" minOccurs="0"/>
          <xs:element name="height" type="gml:LengthType" minOccurs="0"/>
          <xs:element name="distance" type="gml:LengthType" minOccurs="0"/>
          <xs:element name="lod0BaseLine" type="gml:CurvePropertyType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="NoiseCityFurnitureSegment" type="NoiseCityFurnitureSegmentType" substitutionGroup="core:CityObject"/>
</xs:schema>
```

Copyright © 2012 Open Geospatial Consortium.
An example for a feature collection in a corresponding instance document is depicted below. Two CityGML buildings contain application specific properties distinguished from CityGML properties by the namespace prefix `noise`. The other properties, function and geometry, are defined by corresponding CityGML modules. In addition to the buildings, a noise barrier as child of a city furniture element is included in the feature collection. Please note, that the order of the child elements in the sequence is not arbitrary: the child elements defined by an ADE subschema have to occur after the child elements defined by CityGML modules. There is, however, no specific order of the ADE properties.

```xml
...<core:cityObjectMember>
  <bldg:Building gml:id="aa">  
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid>... </bldg:lod1Solid>
    <noise:buildingHabitants>14</noise:buildingHabitants>
    <noise:buildingReflectionCorrection uom="dB">4.123</noise:buildingReflectionCorrection>
  </bldg:Building>
</core:cityObjectMember>
<core:cityObjectMember>
  <bldg:Building gml:id="aaa">
    <bldg:function>1004</bldg:function>
    <bldg:lod1Solid>... </bldg:lod1Solid>
    <noise:buildingReflectionCorrection uom="dB">3.123</noise:buildingReflectionCorrection>
    <noise:buildingHabitants>6</noise:buildingHabitants>
  </bldg:Building>
</core:cityObjectMember>
<core:cityObjectMember>
  <frn:CityFurniture gml:id="CFUR_0815">
    <frn:function>1520</frn:function>
    <frn:lod1Geometry>... </frn:lod1Geometry>
    <noise:noiseCityFurnitureSegmentProperty>
      <noise:NoiseCityFurnitureSegment gml:id="CFRS_0815">
        <noise:reflection>absorbierende Lärmschutzwand</noise:reflection>
        <noise:reflectionCorrection uom="dB">4.123</noise:reflectionCorrection>
        <noise:height uom="m">7.123</noise:height>
        <noise:distance uom="m">21.123</noise:distance>
        <noise:lod0BaseLine>
            <gml:coordinates decimal=." \,\,\, " cs=",\"\,ts=" >5707335,2524175,188 5707338,2524181,188 5707330,2524185,188 5707327,2524179,188 </gml:coordinates>
          </gml:LineString>
        </noise:lod0BaseLine>
    </noise:NoiseCityFurnitureSegment>
  </frn:CityFurniture>
</core:cityObjectMember>
...
10.14 Code lists

CityGML feature types often include attributes whose values can be enumerated in a list of discrete values. An example is the attribute roof type of a building, 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.

If the list of values is fixed, the allowed attribute values are specified in and enforced by the CityGML schema using an enumeration as attribute type. Attributes of an enumerated type may only take values from the predefined list. Examples for such attributes are relativeToTerrain and relativeToWater of the abstract base class core:_CityObject (CityGML Core module, cf. chapter 10.1) as well as wrapMode of the abstract class app:_Texture (Appearance module, cf. chapter 9.4).

In case a fixed enumeration of possible attribute values is not suitable, the attribute type is specified as gml:CodeType and the allowed attribute values can be provided in a code list which is specified outside the CityGML schema. Examples for such attributes are class, function, and usage which are available for almost all CityGML feature types. A code list contains coded attribute values which hinder misspellings and ensure that the same code is used for the same notion or concept. If a code list is provided for an attribute of type gml:CodeType, then any conformant attribute shall only take values from the given code list. This allows applications to validate the attribute values and thus facilitates semantic and syntactic interoperability. The optional codeSpace attribute declared for gml:CodeType is used to associate an attribute with a code list. If a codeSpace is present, its value shall be a persistent URI identifying the code list. If no codeSpace is given, then the attribute value can only be interpreted as a simple text token and validation requires additional knowledge.

The governance of code lists is decoupled from the governance of the CityGML schema and specification. Accordingly, code lists can be specified by any organisation or information community according to their information needs. There shall be one authority per codeSpace and hence per code list who is in charge of the contents and the maintenance of the code list. As a result, the code list values are managed outside the CityGML schema. Thus, in contrast to a fixed enumeration enforced by the CityGML schema, changes to a code list do not require a revision of the CityGML schema and specification.

The contents of code lists may substantially vary for different countries (e.g., due to national law or regulations) and for different information communities. For this reason, this International standard does not specify normative code lists for any of the attributes of type gml:CodeType. However, Annex C provides non-normative code lists for selected attributes which are proposed and maintained by the SIG 3D. These code lists can be directly referenced in CityGML instance documents and serve as an example for the definition of code lists. The code lists given in Annex C comprise the non-normative code lists which are included in the previous version 1.0 of this International standard in order to ensure backwards compatibility.

It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. Whiteside 2005). An example for a code list implemented as simple dictionary is given below. It shows an excerpt of the code list proposed by the SIG 3D for the attribute roofType of the class _AbstractBuilding (Building module, cf. chapter 10.3).

```xml
    <gml:name>roofType</gml:name>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id357">
            <gml:description>flat roof</gml:description>
        </gml:Definition>
    </gml:dictionaryEntry>
    <gml:name>1000</gml:name>
    </gml:Definition>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id358">
            <gml:description>monopitch roof</gml:description>
        </gml:Definition>
    </gml:dictionaryEntry>
    <gml:name>1010</gml:name>
</gml:Dictionary>
```
Listing 2: Example of a code list implemented as simple dictionary following the GML 3.1.1 Simple Dictionary Profile.

In the simple dictionary concept, the code list itself is represented by a *gml:Dictionary* element. The allowed attribute values are listed as *gml:Definition* entries contained in the *gml:Dictionary*. For each definition entry, the coded attribute value is specified by the *gml:name* subelement. Any attribute referencing this code list in a CityGML instance document may only take values which are specified by a *gml:name* element of one of the definition entries. If the attribute value is not specified by one of the definition entries, then the attribute value is invalid. The *gml:description* subelement of a definition entry provides an additional textual description for the coded attribute value. This description can be used, for example, as human readable substitute for the coded attribute value.

The following excerpt of a CityGML instance document illustrates the usage of the code list mechanism. The document contains a *bldg:Building* object whose *roofType* value is taken from the code list shown in Listing 2. The *codeSpace* attribute of the *roofType* element identifies the code list through the globally unique URL http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml which is managed and maintained by the SIG 3D. According to this code list, the coded attribute value 1000 denotes a flat roof for this building.

```xml
<bldg:Building>
  <bldg:roofType
    codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1000</bldg:roofType>
  ...
</bldg:Building>
```
Annex A
(normative)

XML Schema definition

A.1 CityGML Core module

The CityGML Core module is defined within the XML Schema definition file cityGMLBase.xsd. The target namespace http://www.opengis.net/citygml/2.0 is associated with the core module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/2.0" xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
  xmlns:xmldoc="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml"
  targetNamespace="http://www.opengis.net/citygml/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
      CityGML is an OGC Standard. To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
    schemaLocation="http://docs.oasis-open.org/xal/2.0/xAL.xsd"/>
  <xs:complexType name="AbstractCityObjectType" abstract="true">
    <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:annotation>
        <xs:documentation> |
      </xs:annotation>
      <xs:sequence>
        <xs:element ref="GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:extension base="gml:AbstractFeatureCollectionType"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="AbstractCityObjectMemberType" 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. 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"/>
        <xs:element name="terminationDate" type="xs:date" minOccurs="0"/>
        <xs:element name="externalReference" type="ExternalReferenceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="generalizesTo" type="GeneralizationRelationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="relativeToTerrain" type="RelativeToTerrainType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="relativeToWater" type="RelativeToWaterType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="CityModelType">
    <xs:annotation>
      <xs:documentation> |
    </xs:annotation>
    <xs:complexContent>
      <xs:annotation>
        <xs:documentation> |
      </xs:annotation>
      <xs:sequence>
        <xs:element ref="GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:extension base="gml:_FeatureCollection" substitutionGroup="gml:_Feature"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="_GenericApplicationPropertyOfCityModel" type="xs:anyType" abstract="true" substitutionGroup="gml:_FeaturePropertyType">
    <xs:annotation>
      <xs:documentation> |
    </xs:annotation>
    <xs:complexContent>
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCityObject" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="_GenericApplicationPropertyOfCityObject" type="xs:anyType" abstract="true" substitutionGroup="gml:_Feature">
    <xs:annotation>
      <xs:documentation> |
    </xs:annotation>
    <xs:complexContent>
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCityModel" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```
<xs:simpleType>  
<xs:restriction base="gml:AbstractCityObjectType" type="xs:anyType" abstract="true">  
<xs:choice>  
<xs:element name="_Site" type="gml:AbstractSiteType" abstract="true" substitutionGroup="_CityObject"/>  
</xs:choice>  
</xs:restriction>  
</xs:simpleType>

<xs:simpleType>  
<xs:restriction base="gml:AbstractCityObjectType" type="xs:anyType" abstract="true">  
<xs:choice>  
<xs:element name="_GenericApplicationPropertyOfSite" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
<xs:element name="_GenericApplicationPropertyOfSite" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:choice>  
</xs:restriction>  
</xs:simpleType>

<xs:simpleType>  
<xs:restriction base="gml:AbstractCityObjectType" type="xs:anyType" abstract="true">  
<xs:choice>  
<xs:element name="_Site" type="gml:AbstractSiteType" abstract="true" substitutionGroup="_CityObject"/>  
<xs:element name="_GenericApplicationPropertyOfSite" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:choice>  
</xs:restriction>  
</xs:simpleType>

<xs:complexType name="_GenericApplicationPropertyOfSite">  
<xs:complexContent base="gml:ApplicationPropertyOfCityObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_CityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="_CityObject">  
<xs:complexContent base="gml:CityObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_CityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="_Site">  
<xs:complexContent base="gml:Site">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_Site"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="_CityObject">  
<xs:complexContent base="gml:CityObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_CityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="ExternalObjectReferenceType">  
<xs:complexContent base="gml:AbstractObjectReferenceType">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element name="informationSystem" type="xs:anyURI" minOccurs="0"/>
<xs:element name="externalObject" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="_GenericApplicationPropertyOfSite">  
<xs:complexContent base="gml:ApplicationPropertyOfCityObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_GenericApplicationPropertyOfSite" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="GeneralizationRelationType">  
<xs:complexContent base="gml:AbstractRelationType">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_GenericApplicationPropertyOfSite" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="AbstractSiteType">  
<xs:complexContent base="gml:Site">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_Site"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="AbstractCityObjectType">  
<xs:complexContent base="gml:CityObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_CityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="AbstractCityObjectRealizationType">  
<xs:complexContent base="gml:CityObjectRealization">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_CityObjectRealization"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:complexType name="AbstractApplicationPropertyOfObject">  
<xs:complexContent base="gml:ApplicationPropertyOfObject">  
<xs:sequence minOccurs="0" maxOccurs="unbounded">  
<xs:element ref="_GenericApplicationPropertyOfObject" type="gml:AbstractApplicationPropertyOfObject" abstract="true" substitutionGroup="gml:AbstractApplicationPropertyOfCityObject"/>  
</xs:sequence>  
</xs:complexContent>  
</xs:complexType>

<xs:simpleType name="RelativeToTerrainType">  
<xs:restriction base="gml:GML3.AnyURI" type="xs:anyURI" abstract="false">  
<xs:enumeration name="entirelyAboveTerrain" value="entirelyAboveTerrain"/>  
<xs:enumeration name="aboveTerrain" value="aboveTerrain"/>  
<xs:enumeration name="substantiallyAboveAndBelowTerrain" value="substantiallyAboveAndBelowTerrain"/>  
<xs:enumeration name="substantiallyAboveTerrain" value="substantiallyAboveTerrain"/>  
<xs:enumeration name="substantiallyBelowTerrain" value="substantiallyBelowTerrain"/>  
<xs:enumeration name="substantiallyBelowAndAboveTerrain" value="substantiallyBelowAndAboveTerrain"/>  
<xs:enumeration name="substantiallyBelowTerrain" value="substantiallyBelowTerrain"/>  
<xs:enumeration name="entirelyBelowTerrain" value="entirelyBelowTerrain"/>  
<xs:enumeration name="belowTerrain" value="belowTerrain"/>  
</xs:restriction>  
</xs:simpleType>
th even a projection might be modelled by the transformation matrix. The concept of implicit geometries is an enhancement of is a 4x4 matrix that is multiplied with the prototype coordinates using homogeneous coordinates, i.e. (x,y,z,1). This way results. The transformation matrix accounts for the intended rotation, scaling, and local translation of the prototype. It absolute coordinates of an implicit geometry, the anchor point coordinates have to be added to the matrix multiplication an anchor point denoting the base point of the object in the world coordinate reference system. In order to determine th the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local cart sign. This prototypic geometry object is re 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).</xs:documentation>
</xs:annotation>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
</xs:complexType>
<xs:complexType name="ImplicitRepresentationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a CityObject to its implicit geometry representation, which is a representation of a geometry by referencing a prototype and transforming it to its real position in space. The ImplicitRepresentationPropertyType element must either carry a reference to a ImplicitGeometry object or contain a ImplicitGeometry object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="ImplicitGeometry"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>

<xs:simpleType name="doubleBetween0and1">
  <xs:annotation>
    <xs:documentation>Type for 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:restriction base="xs:double">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="1"/>
  </xs:restriction>
</xs:simpleType>

<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="TransformationMatrix4x4Type">
  <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:length value="16"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TransformationMatrix2x2Type">
  <xs:annotation>
    <xs:documentation>Used for georeferencing. The Transformation matrix is a 2 by 2 matrix, thus it must be a list with 4 items. The order the matrix element are represented is row-major, i.e. the first 2 elements represent the first row, the fifth to the eight element the second row,... </xs:documentation>
  </xs:annotation>
  <xs:restriction base="gml:doubleList">
    <xs:length value="4"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TransformationMatrix3x4Type">
  <xs:annotation>
    <xs:documentation>Used for texture parameterization. The Transformation matrix is a 3 by 4 matrix, thus it must be a list with 12 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:length value="12"/>
  </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:int"/>
</xs:simpleType>
<xs:minInclusive value="0"/>
<xs:maxInclusive value="4"/>
</xs:restriction>
</xs:simpleType>
</xs:schema>
A.2 Appearance module

The CityGML Appearance module is defined within the XML Schema definition file appearance.xsd. The target namespace http://www.opengis.net/citygml/appearance/2.0 is associated with this extension module.

```xml
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" targetNamespace="http://www.opengis.net/citygml/appearance/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.0.0"/>
```

CityGML is an OGC Standard.

Copyright (c) 2012 Open Geospatial Consortium.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal.
```xml
<xs:element name="Appearance" type="AppearanceType" substitutionGroups="gml:_Feature"/>  
<xs:element name="appearanceMember" type="gml:FeaturePropertyType" substitutionGroups="gml:featureMember"/>  
<xs:element name="appearance" type="AppearancePropertyType" substitutionGroups="core:_GenericApplicationPropertyOfCityObject"/>  
<xs:complexType name="AbstractSurfaceDataType" abstract="true"/>
```

```
<xs:element name="SurfaceData" type="AbstractSurfaceDataType" abstract="true" substitutionGroups="gml:_Feature"/>
```

```xml
...</xs:schema>
```

Copyright © 2012 Open Geospatial Consortium.
<xsd:element name="_GenericApplicationPropertyOfSurfaceData" type="xs:anyType" abstract="true"/>
</xsd:complexType>
</xsd:annotation>
</xsd:sequence>
</xsd:complexType>
</xsd:documentation>
Denotes the relation of an Appearance to its surface data. The SurfaceDataPropertyType element must either carry a reference to a SurfaceData object or contain a SurfaceData object inline, but neither both nor none.
</xsd:documentation>
</xsd:sequence>
</xsd:complexType>
</xsd:annotation>
</xsd:element ref="_SurfaceData" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:documentation>
Base class for textures. "imageURI" can contain any valid URI from references to a local file to preformatted web service requests. The linking to geometry and texture parameterization is provided by derived classes.
</xsd:documentation>
</xsd:complexType>
</xsd:extension>
</xsd:complexType>
</xsd:element name="_GenericApplicationPropertyOfParameterizedTexture">
<!----------------------------------------------------------------------------------------------------------------------------------------->
</xsd:complexType>
</xsd:element name="_GenericApplicationPropertyOfParameterizedTexture" type="xs:anyType" abstract="true"/>
</xsd:complexType>
</xsd:annotation>
</xsd:sequence>
</xsd:complexType>
</xsd:annotation>
</xsd:element name="_Texture" type="AbstractTextureType" abstract="true" substitutionGroup="_SurfaceData"/>
<!----------------------------------------------------------------------------------------------------------------------------------------->
</xsd:element name="ParameterizedTextureType">
<!----------------------------------------------------------------------------------------------------------------------------------------->
</xsd:element name="ParameterizedTexture" type="ParameterizedTextureType" substitutionGroup="_Texture"/>
<!----------------------------------------------------------------------------------------------------------------------------------------->
</xsd:element name="GeoreferencedTextureType">
<!----------------------------------------------------------------------------------------------------------------------------------------->
</xsd:complexType>
</xsd:documentation>
Specialization for georeferenced textures, i.e. textures using a planimetric projection. Such textures contain an implicit parameterization (either stored within the image file, in an accompanying world file, or using the
"referencePoint" and "orientation"-elements). A georeference provided by "referencePoint" and "orientation" always takes precedence. The search order for an external georeference is determined by the boolean flag preferWorldFile. If this flag is set to true (its default value), a world file is looked for first and only if it is not found the georeference from the image data is used. If preferWorldFile is false, the world file is used only if no georeference from the image data is available. The "boundedBy"-property should contain the bounding box of the projected image data. Since a georeferenced texture has a unique parameterization, "target" only provides links to surface geometry without any additional texture parameterization. Only gml:MultiSurface or descendents of gml:AbstractSurfaceType are valid targets.

```xml
<xs:complexType
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:annotation>
        <xs:documentation>
            Denotes the relation of a texture to a surface, that is augmented by a TextureParameterization object. The TextureAssociationType element must either carry a reference to a _TextureParameterization object or contain a transformation matrix are available. Currently, texture coordinates and texture coordinate generation using a transformation matrix are available. 
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:restriction base="AbstractTextureType">
            <xs:sequence>
                <xs:element name="preferWorldFile" type="xs:boolean" default="true" minOccurs="0"/>
                <xs:element name="referencePoint" type="gml:PointPropertyType" minOccurs="0"/>
                <xs:element name="target" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>

```

---

Copyright © 2012 Open Geospatial Consortium.
class for defining constant surface properties. It is based on X3D's material definition. In addition, "isSmooth" provides a hint for value interpolation. The link to surface geometry is established via the "target"-property. Only gml:MultiSurface or descendants of gml:AbstractSurfaceType are valid targets.

Textures can be qualified by the attribute textureType. The textureType differentiates between textures, which are specific for a certain object and are only used for that object (specific), and prototypic textures being typical for that kind of object and are used many times for all objects of that kind (typical). A typical texture may be replaced by a specific, if available. Textures may also be classified as unknown.
<xs:simpleType name="Color">
  <xs:documentation>List of three values (red, green, blue), separated by spaces. The values must be in the range between zero and one.</xs:documentation>
  <xs:restriction base="core:doubleBetween0and1List">
    <xs:length value="3"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ColorPlusOpacity">
  <xs:documentation>List of three or four values (red, green, blue, opacity), separated by spaces. The values must be in the range between zero and one. If no opacity is given, it is assumed as 1.0.</xs:documentation>
  <xs:restriction base="core:doubleBetween0and1List">
    <xs:minLength value="3"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
A.3 Bridge module

The CityGML Bridge module is defined within the XML Schema definition file bridge.xsd. The target namespace http://www.opengis.net/citygml/bridge/2.0 is associated with this extension module.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/bridge/2.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/citygml/bridge/2.0" xmlns:gml="http://www.opengis.net/gml"
  schemaLocation="bridge.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified"
  version="2.0.0"/>
  <xs:annotation>
    <xs:documentation>
      Type describing the thematic and geometric attributes and the associations of bridges. It is an abstract type, only its subclasses Bridge and BridgePart can be instantiated. An _AbstractBridge may consist of BridgeParts, which are again _AbstractBridges by inheritance. Thus an aggregation hierarchy between _AbstractBridges of arbitrary depth may be specified. In such an hierarchy, top elements are Bridges, while all other elements are BridgeParts. Each element of such a hierarchy may have all attributes and geometries of _AbstractBridges. It must, however, be assured that no inconsistencies occur.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
      <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
      <xs:element name="isMovable" type="xs:boolean" default="false" minOccurs="0"/>
<xs:complexType name="BridgeType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridge" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgePartType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgePartType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfBridgePart" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgeInstallationType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgeInstallationType">
      <xs:sequence>
        <xs:element ref="/" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgeInstallationPropertyType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgeInstallationPropertyType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="lod3Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="0"/>
        <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgeType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgeType">
      <xs:sequence>
        <xs:element ref="/" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgePartType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgePartType">
      <xs:sequence>
        <xs:element ref="/" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BridgeInstallationType">
  <xs:complexContent>
    <xs:restriction base="AbstractBridgeInstallationType">
      <xs:sequence>
        <xs:element ref="/" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
An IntBridgeInstallation is an interior part of a Bridge which has a specific function or semantic meaning. Examples are interior stairs, railings, radiators or pipes. As subclass of _CityObject, an IntBridgeInstallation inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations. 

Denotes the relation of an _AbstractBridge to its interior bridge installations. The IntBridgeInstallationPropertyType element must either carry a reference to a IntBridgeInstallation object or contain a IntBridgeInstallation object inline, but neither both nor none.

A BridgeConstructionElement is a part of a Bridge which has not the significance of a BridgePart. In contrast to BridgeInstallation, a BridgeConstructionElement is essential from a structural point of view. Examples are pylons, anchorages, etc. As subclass of _CityObject, a BridgeInstallation inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.
AbstractSurfaceType is defined in ISO 19107 as a surface which classifies surfaces bounding an AbstractBridge, BridgeInstallation, InBuildingInstallation, BridgeConstructionElement, and BridgeRoom. 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.

BridgeConstructionElementPropertyType element must either carry a reference to a BridgeConstructionElement object or contain a BridgeConstructionElement object inline, but neither both nor none.

The BridgeConstructionElementPropertyType element contains a BridgeConstructionElementPropertyType element which can either contain a BridgeConstructionElement object or a reference to a BridgeConstructionElement object.

BoundarySurfaceType is a thematic object which classifies surfaces bounding an AbstractBridge, BridgeInstallation, InBuildingInstallation, BridgeConstructionElement, and BridgeRoom. 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.

A BoundarySurface is thematic and classifies surfaces bounding an AbstractBridge, BridgeInstallation, InBuildingInstallation, BridgeConstructionElement, and BridgeRoom. 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="InteriorWallSurface" type="InteriorWallSurfaceType" substitutionGroup="_BoundarySurface"/>
</xs:complexType>

<xs:complexType name="GenericApplicationPropertyOfInteriorWallSurface">
  <xs:complexContent>
    <xs:extension base="AbstractApplicationPropertyType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BoundarySurfacePropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _AbstractBridge to its bounding thematic surfaces (walls, roofs, ...). The
      BoundarySurfacePropertyType element must either carry a reference to a _BoundarySurface object or contain a
      _BoundarySurface object inline, but neither both nor none. There is no differentiation between interior surfaces bounding
      rooms and outer ones bounding bridges (one reason is, that ClosureSurface belongs to both types). It has to be made sure
      by additional integrity constraints that, e.g. an _AbstractBridge is not related to CeilingSurfaces or a room not to
      RoofSurfaces. </xs:documentation>
  </xs:annotation>

  <xs:element name="BoundarySurface" type="_BoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="OpeningPropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _BoundarySurface to its openings (doors, windows). The OpeningPropertyType
      element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor
      none. </xs:documentation>
  </xs:annotation>

  <xs:element name="Opening" type="_Opening" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="AbstractOpeningType" abstract="true"/>

<xs:annotation>
  <xs:documentation>
    Type for openings (doors, windows) in boundary surfaces. 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:complexType name="CoreAbstractCityObjectOfType">
  <xs:annotation>
    <xs:documentation>
      Type for openings (doors, windows) in boundary surfaces. 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:complexType>

Copyright © 2012 Open Geospatial Consortium.
A BridgeRoom is a thematic object for modelling the closed parts inside a Bridge. 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 BridgeRoom 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 BridgeRoom are referenced by the boundedBy property. As subclass of _CityObject, a BridgeRoom inherits all attributes and relations, in particular an id, names, external references, generic attributes and generalization relations.

Type for doors in boundary surfaces. 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:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="lod4Geometry" type="gml:GeometryPropertyType" minOccurs="0"/>
<xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>
<xs:element ref="_GenericApplicationPropertyOfBridgeFurniture" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="BridgeFurniture" type="BridgeFurnitureType" substitutionGroup="core:_CityObject"/>
<xs:element name="_GenericApplicationPropertyOfBridgeFurniture" type="xs:anyType" abstract="true"/>

<xs:complexType name="InteriorBridgeRoomPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of an AbstractBridge to its rooms. The InteriorBridgeRoomPropertyType element must either carry a reference to an BridgeRoom object or contain an BridgeRoom object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="BridgeRoom"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>

<xs:complexType name="InteriorFurniturePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a BridgeRoom to its interior bridge furniture. The InteriorBridgeFurniturePropertyType element must either carry a reference to an BridgeFurniture object or contain an BridgeFurniture object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:sequence minOccurs="0">
    <xs:element ref="BridgeFurniture"/>
  </xs:sequence>
  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>
A.4 Building module

The CityGML Building module is defined within the XML Schema definition file building.xsd. The target namespace http://www.opengis.net/citygml/building/2.0 is associated with this extension module.

```xml
<xs:schema namespace="http://www.opengis.net/citygml/building/2.0" xmlns="http://www.opengis.net/citygml/building/2.0" targetNamespace="http://www.opengis.net/citygml/building/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.0.0"/>
```

This module Type describing the thematic and geometric attributes and the associations of buildings. It is an abstract type, only its subclasses Building and BuildingPart can be instantiated. An AbstractBuilding may consist of arbitrary depth may be specified. In such an hierarchy, top elements are Buildings, while all other elements are BuildingParts. Each element of such a hierarchy may have all attributes and geometries of AbstractBuildings. It must, however, be assured than no inconsistencies occur (for example, if the geometry of a Building does not correspond to the geometries of its parts, or if the roof type of a Building is saddle roof, while its parts have an hip roof). As subclass of CityObject, an AbstractBuilding inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

```xml
<xs:complexType name="AbstractBuildingType" abstract="true">
  <xs:annotation>
    <xs:documentation>
      Type describing the thematic and geometric attributes and the associations of buildings. It is an abstract type, only its subclasses Building and BuildingPart can be instantiated. An AbstractBuilding may consist of arbitrary depth may be specified. In such an hierarchy, top elements are Buildings, while all other elements are BuildingParts. Each element of such a hierarchy may have all attributes and geometries of AbstractBuildings. It must, however, be assured than no inconsistencies occur (for example, if the geometry of a Building does not correspond to the geometries of its parts, or if the roof type of a Building is saddle roof, while its parts have an hip roof). As subclass of CityObject, an AbstractBuilding inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
    </xs:documentation>
  </xs:annotation>

  <xs:complexContent>
    <xs:extension base="core:AbstractBuildingType" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="roofType" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="measuredHeight" type="gml:LengthType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="storeysBelowGround" type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="storeysAboveGround" type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0RoofEdge" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4RoofEdge" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod0Terrain" type="gml:MultiCurvePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod1Roof" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod2Terrain" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod3Roof" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lod4Roof" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:schema>
```
<xs:complexType>
    <xs:complexContent>
        <xs:extension base="AbstractBuildingType"> 
            <xs:sequence>
                <xs:element ref="_GenericApplicationPropertyOfBuildingType" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="BuildingInstallationType">
    <xs:annotation>
        <xs:documentation>
            A BuildingInstallation 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, and generalization relations. 
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="AbstractCityObjectType"> 
            <xs:sequence>
                <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                <xs:element name="interiorRoom" type="InteriorRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="consistsOfBuildingPart" type="BuildingPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType>
  <xs:complexContent>
    <xs:extension base="core:AbstractCityObjectType">
      <xs:sequence>
        <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="lodMultiSurface" type="gml:MultiSurfaceType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="boundedBy" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="_GenericApplicationPropertyOfAbstractBuildingInstallation" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="OuterFloorSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="InteriorWallSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfInteriorWallSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="CeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="OuterCeilingSurfaceType">
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfOuterCeilingSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="BoundarySurfacePropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _AbstractBuilding to its bounding thematic surfaces (walls, roofs, ...). The BoundarySurfacePropertyType element must either carry a reference to a _BoundarySurface object or contain a BoundarySurface object inline, but neither both nor none. There is no differentiation between interior surfaces bounding rooms and outer ones bounding buildings (one reason is, that ClosureSurface belongs to both types). It has to be made sure by additional integrity constraints that, e.g. an _AbstractBuilding is not related to CeilingSurfaces or a room not to RoofSurfaces. </xs:documentation>
  </xs:annotation>
</xs:complexType>

<xs:complexType name="OpeningPropertyType">
  <xs:annotation>
    <xs:documentation>
      Denotes the relation of an _BoundarySurface to its openings (doors, windows). The OpeningPropertyType element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
</xs:complexType>
rooms are connected if both use the same opening object or the same closure surface. The thematic surfaces bounding a room topologically connected. Necessary by using closure surfaces. The geometry may be either a solid, or a MultiSurface if the boundary is not a solid. As subclass of _CityObject, an _Opening inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

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 boundedBy property. As subclass of _CityObject, a Room inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
particular an id, names, external references, and generalization relations. 

Type for building furnitures. As subclass of _CityObject, a BuildingFurniture inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Denotes the relation of an _AbstractBuilding to its rooms. The InteriorRoomPropertyType element must either carry a reference to a Room object or contain a Room object inline, but neither both nor none.

Denotes the relation of a Room to its interior furnitures (movable). The InteriorFurniturePropertyType element must either carry a reference to a BuildingFurniture object or contain a BuildingFurniture object inline, but neither both nor none.
A.5 CityFurniture module

The CityGML CityFurniture module is defined within the XML Schema definition file cityFurniture.xsd. The target namespace http://www.opengis.net/citygml/cityfurniture/2.0 is associated with this extension module.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema

<xs:complexType name="CityFurnitureType">

<xs:complexContent>

<xs:extension base="core:AbstractCityObjectType">

<xs:sequence>

<xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>

<xs:element name="usage" type="gml:CodeType" 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="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>

<xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>

<xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>

<xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>

<xs:element name="lod1ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>

<xs:element name="lod2ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>

<xs:element name="lod3ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>

<xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0"/>

<xs:element ref="_GenericApplicationPropertyOfCityFurniture" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>
```
A.6 CityObjectGroup module

The CityGML CityObjectGroup module is defined within the XML Schema definition file cityObjectGroup.xsd. The target namespace http://www.opengis.net/citygml/cityobjectgroup/2.0 is associated with this extension module.

```xml
<xs:schema xmlns="http://www.opengis.net/citygml/cityobjectgroup/2.0" xmlns:core="http://www.opengis.net/citygml/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" targetNamespace="http://www.opengis.net/citygml/cityobjectgroup/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.0.0"/>
```

CityGML is an OGC Standard.
Copyright (c) 2012 Open Geospatial Consortium.
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

```xml
<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, and generalization relations. As CityObjectGroup itself is a CityObject, it may also be contained by another group. </xs:documentation>
  </xs:annotation>
</xs:complexType>
```

```xml
<xs:complexType name="CityObjectGroupMemberType">
  <xs:annotation>
    <xs:documentation> Denotes the relation of a CityObjectGroup to its members, which are _CityObjects. The CityObjectGroupMemberType element must either carry a reference to a _CityObject object or contain a _CityObject object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
</xs:complexType>
```

```xml
<xs:complexType name="CityObjectGroupParentType">
  <xs:annotation>
    <xs:documentation> Denotes the relation of a CityObjectGroup to its parent, which is a CityObject. The CityObjectGroupParentType element must either carry a reference to a _CityObject object or contain a _CityObject object inline, but neither both nor none. The parent association allows for modelling of a generic hierarchical grouping concept. Named aggregations of components (CityObjects) can be added to specific CityObjects considered as the parent object. The parent association links to the aggregate, while the parts are given by the group members. </xs:documentation>
  </xs:annotation>
</xs:complexType>
```
A.7 Generics module

The CityGML Generics module is defined within the XML Schema definition file generics.xsd. The target namespace http://www.opengis.net/citygml/generics/2.0 is associated with this extension module.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/generics/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
    targetNamespace="http://www.opengis.net/citygml/generics/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="2.0.0">
  <xs:documentation>
    CityGML is an OGC Standard.
    Copyright (c) 2012 Open Geospatial Consortium.
    To obtain additional rights of use, visit http://www.opengeospatial.org/legal.
  </xs:documentation>
  <xs:import namespace="http://www.opengis.net/citygml/2.0" schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <xs:complexType name="GenericCityObjectType">
    <xs:documentation>
      Generic (user defined) city objects may be used to model features which are not covered explicitly by the CityGML schema. Generic objects must be used with care; they shall only be used if there is no appropriate thematic class available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 
    </xs:documentation>
    <xs:annotation>
      <xs:documentation>
        Generic (user defined) city objects may be used to model features which are not covered explicitly by the CityGML schema. Generic objects must be used with care; they shall only be used if there is no appropriate thematic class available in the overall CityGML schema. Otherwise, problems concerning semantic interoperability may arise. As subclass of _CityObject, a generic city object inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 
      </xs:documentation>
      <xs:documentation>
        Generics module
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:restriction base="core:_CityObject">
        <xs:sequence>
          <xs:element name="GenericCityObject" type="GenericCityObjectType" substitutionGroup="core:_CityObject"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```
<xs:documentation/>
</xs:annotation>
<xs:complexType base="AbstractGenericAttributeType">
  <xs:sequence>
    <xs:element name="value" type="gml:MeasureType"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="measureAttribute" type="MeasureAttributeType" substitutionGroup="_genericAttribute"/>

<xs:complexType name="GenericAttributeSetType">
  <xs:annotation>
    <xs:documentation>
      Set of generic attributes with an optional codeSpace. If the codeSpace attribute is present, then its value should identify an authority for the set, such as the organisation or community who defined its content. The generic attribute set may contain arbitrary generic attributes.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractGenericAttributeType">
      <xs:sequence>
        <xs:element ref="_genericAttribute" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="genericAttributeSet" type="GenericAttributeSetType" substitutionGroup="_genericAttribute"/>

</xs:schema>
A.8 LandUse module

The CityGML LandUse module is defined within the XML Schema definition file landUse.xsd. The target namespace http://www.opengis.net/citygml/landuse/2.0 is associated with this extension module.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/landuse/2.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/citygml/landuse/2.0"
  version="2.0.0">
  <xs:annotation>
    <xs:documentation>
      CityGML is an OGC Standard.
      Copyright (c) 2012 Open Geospatial Consortium.
      To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.
    </xs:documentation>
  </xs:annotation>
  <xs:import
    namespace="http://www.opengis.net/gml"
    schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <xs:import
    namespace="http://www.opengis.net/citygml/2.0"
    schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <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, and generalization relations.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod0MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="_GenericApplicationPropertyOfLandUse" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="LandUse" type="LandUseType" substitutionGroup="core:_CityObject"/>
  <xs:element name="_GenericApplicationPropertyOfLandUse" type="xs:anyType" abstract="true"/>
</xs:schema>
A.9 Relief module

The CityGML Relief module is defined within the XML Schema definition file relief.xsd. The target namespace http://www.opengis.net/citygml/relief/2.0 is associated with this extension module.

```xml
<xs:schema xmlns="http://www.opengis.net/citygml/relief/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
xmlns:gml="http://www.opengis.net/gml"
targetNamespace="http://www.opengis.net/citygml/relief/2.0" elementFormDefault="qualified" attributeFormDefault="unqualified" version="2.0.0">
  <xs:annotation>
    <xs:documentation>
      CityGML is an OGC Standard. To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase.xsd"/>
  <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, and generalization relations.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:sequence>
        <xs:element name="lod" type="core:integerBetween0and4"/>
        <xs:element name="reliefComponent" type="ReliefComponentPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="AbstractReliefComponentType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="AbstractReliefComponentType" 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, and generalization relations.</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:sequence>
        <xs:element name="lod" type="core:integerBetween0and4"/>
        <xs:element name="extent" type="gml:PolygonPropertyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="AbstractReliefComponentType" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ReliefComponentPropertyType">
    <xs:annotation>
      <xs:documentation>Denotes the relation of a ReliefFeature to its components. The ReliefComponent PropertyType element must either carry a reference to a _ReliefComponent object or contain a _ReliefComponent object inline, but neither both nor none. </xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="ReliefComponent"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```
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, and generalization relations.

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, and generalization relations.

Type describing the mass point component of a relief feature. As subclass of _CityObject, a MassPointRelief inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Type describing the break line Component of a relief feature. A break line relief consists of break lines or ridgeOrValleyLines. As subclass of CityObject, a BreaklineRelief inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
<xs:complexType name="BreaklineReliefType" substitutionGroup="_ReliefComponent"/>

<!--=========================================================================================-->

<xs:element name="_GenericApplicationPropertyOfBreaklineRelief" type="xs:anyType" abstract="true"/>

<!--=========================================================================================-->

<xs:complexType name="tinPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a TINRelief to its components. The tinPropertyType element must either carry a reference to a gml:TriangulatedSurface object or contain a gml:TriangulatedSurface object inline, but neither both nor none. <xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:TriangulatedSurface"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
</xs:complexType>

<!--=========================================================================================-->

<xs:complexType name="gridPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a RasterReliefType to its components. The gridPropertyType element must either carry a reference to a gml:RectifiedGridCoverage object or contain a gml:RectifiedGridCoverage object inline, but neither both nor none. <xs:documentation>
    </xs:annotation>
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:RectifiedGridCoverage"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
</xs:complexType>

<!--=========================================================================================-->

<xs:element name="Elevation" type="gml:LengthType" substitutionGroup="_Object"/>
</xs:schema>
A.10 Transportation module

The CityGML Transportation module is defined within the XML Schema definition file transportation.xsd. The target namespace http://www.opengeospatial.org/ogc/ is associated with this extension module.

```xml
    xmlns:cts="http://www.opengis.net/citygml/transportation/2.0"
    xmlns:coreGML="http://www.opengis.net/gml"
    targetNamespace="http://www.opengis.net/citygml/transportation/2.0"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified"
    version="2.0.0"/>
</xs:schema>
```

CityGML is an OGC Standard.

To obtain additional rights of use, visit http://www.opengeospatial.org/legal.

A Transportation Complex inherits all attributes and relations, in particular an id, names, external references, and 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, and generalization relations.

A Traffic Area inherits all attributes and relations, in particular an id, names, external references, and parts (traffic areas, e.g. pedestrian path, and auxiliary traffic areas). As subclass of _CityObject, a TrafficArea inherits all attributes and relations, in particular an id, names, external references, and basic geometric representation.

As subclass of _CityObject, a TrafficArea inherits all attributes and relations, in particular an id, names, external references, and basic geometric representation.
TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object subclass of _CityObject, aakes the relation of TransportationComplex to its parts, which are traffic areas. The TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object inline, but neither both nor none. 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, and generalization relations.

TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object subclass of _CityObject, aakes the relation of TransportationComplex to its parts, which are traffic areas. The TrafficAreaPropertyType element must either carry a reference to a TrafficArea object or contain a TrafficArea object inline, but neither both nor none. 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, and generalization relations.
_CityObject, a Track inherits all attributes and relations, in particular an id, names, external references, and generalization relations.  

Type describing the class for roads. As subclass of _CityObject, a Road inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 

Type describing the class for railways. As subclass of _CityObject, a Railway inherits all attributes and relations, in particular an id, names, external references, and generalization relations. 

Type describing the class for squares. A square is an open area commonly found in cities (like a plaza). As subclass of _CityObject, a Square inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
A.11 Tunnel module

The CityGML Tunnel module is defined within the XML Schema definition file tunnel.xsd. The target namespace http://www.opengis.net/citygml/tunnel/2.0 is associated with this extension module.

```xml
<xs:schema

<!DOCTYPE xs:schema SYSTEM "tunnel.xsd">
<xs:schema

<xs:element name="Tunnel" type="xs:anyType" abstract="true" substitutionGroup="core:Site"/>
<xs:element name="_GenericApplicationPropertyOfAbstractTunnel" type="xs:anyType" abstract="true"/>
<xs:element name="AbstractTunnel" type="AbstractTunnelType" abstract="true" substitutionGroup="core:_Site"/>
<xs:element name="_AbstractTunnel" abstract="true" substitutionGroup="core:_Site"/>
<xs:element name="TunnelInstallation" type="TunnelInstallationType" abstract="true" substitutionGroup="core:_Site" />
<xs:element name="interiorTunnelInstallation" type="AbstractTunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="intTunnelInstallationProperty" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="interiorHollowSpace" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="intTunnelInstallation" type="TunnelInstallationType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="interiorTunnelInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="TunnelInstallationProperty" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="interiorHollowSpace" type="TunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="interiorTunnelInstallation" type="AbstractTunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="_AbstractApplicationPropertyOfAbstractTunnel" type="xs:anyType" abstract="true"/>
```
Copyright © 2012 Open Geospatial Consortium.
Enforce the requirement that InteriorTunnelInstallationPropertyType elements are not allowed to cross thematic surfaces. The InteriorTunnelInstallationPropertyType element must either carry a reference to an IntTunnelInstallation object or contain a IntTunnelInstallation object inline, but neither both nor none. Alternatively, it is allowed to inherit the IntTunnelInstallation object from its parent parent object in a containment tree. In the above cases, there is no differentiation between interior surfaces bounding hollow spaces and outer ones bounding tunnels (one reason is, that ClosureSurface belongs to both types). It has to be made sure by additional integrity constraints that, e.g. a tunnel is not related to CeilingSurfaces or a room not to RoofSurfaces.

Abstract super class of the features RoofSurface, WallSurface, GroundSurface, ClosureSurface, FloorSurface, OuterFloorSurface, CeilingSurface and OuterCeilingSurface. For each feature, there is a specific feature type that is derived from this abstract type. For each feature, there is a specific feature type that is derived from this abstract type.
<xs:complexType name="RoofSurfaceType">
  <xs:annotation>
    <xs:documentation>Construction that separates the interior part of the Tunnel from the ambient medium (water, air, ..) from above. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType" type="RoofSurfaceType" substitutionGroup="_BoundarySurface"/>
    <xs:element ref="_GenericApplicationPropertyOfRoofSurface" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="WallSurfaceType">
  <xs:annotation>
    <xs:documentation>Mainly vertical construction that separates the interior part of the Tunnel from the ambient medium (rock, earth, water, air, ..) from below on the lowest level. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType" type="WallSurfaceType" substitutionGroup="_BoundarySurface"/>
    <xs:element ref="_GenericApplicationPropertyOfWallSurface" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="GroundSurfaceType">
  <xs:annotation>
    <xs:documentation>Horizontal construction that separates the interior part of the Tunnel from the ambient medium (rock, earth, water, air, ..) from below on the lowest level. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType" type="GroundSurfaceType" substitutionGroup="_BoundarySurface"/>
    <xs:element ref="_GenericApplicationPropertyOfGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="ClosureSurfaceType">
  <xs:annotation>
    <xs:documentation>Virtual surface which can be used to define the volume of geometric objects being not totally bounded by real surfaces (e.g. the entrance of an open Tunnel). </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType" type="ClosureSurfaceType" substitutionGroup="_BoundarySurface"/>
    <xs:element ref="_GenericApplicationPropertyOfClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="FloorSurfaceType">
  <xs:annotation>
    <xs:documentation>Mostly horizontal construction that bounds a HollowSpace from below. </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractBoundarySurfaceType" type="FloorSurfaceType" substitutionGroup="_BoundarySurface"/>
    <xs:element ref="_GenericApplicationPropertyOfFloorSurface" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>
Mainly horizontal construction that separates the interior part of the Tunnel from the ambient medium (rock, earth, water, air, ...) from below. 

Mostly vertical construction that bounds a HollowSpace.

Mostly horizontal construction that bounds a HollowSpace from above.

Horizontal construction that separates the interior part of the Tunnel from the ambient medium (rock, earth, water, air, ...) from above.
<xs:extension>
  <xs:complexType>
    <xs:element name="OuterCeilingSurface" type="OuterCeilingSurfaceType" substitutionGroup="_BoundarySurface"/>
  </xs:complexType>
</xs:extension>

<xs:extension>
  <xs:complexType>
    <xs:element name="GenericApplicationPropertyOfOuterCeilingSurface" type="xs:anyType" abstract="true"/>
  </xs:complexType>
</xs:extension>

<xs:complexType name="HollowSpaceType">
  <xs:annotation>
    <xs:documentation>Area or volume within a Tunnel bounded actually or theoretically </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:sequence base="core:AbstractCityObjectType">
      <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lod" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="lodSolid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="hollowSpaceInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="hollowSpaceInstallation" type="IntTunnelInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="interiorFurniture" type="InteriorFurniturePropertyType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="TunnelFurnitureType">
  <xs:annotation>
    <xs:documentation>Movable, functional objects, whether useful or ornamental, usually found in a HollowSpace</xs:documentation>
  </xs:annotation>
  <xs:complexContent base="core:AbstractCityObjectType">
    <xs:element name="class" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="lod4ImplicitRepresentation" type="core:ImplicitRepresentationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="_GenericApplicationPropertyOfTunnelFurniture" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="InteriorHollowSpacePropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a Tunnel to its internal hollow spaces. The InteriorHollowSpacePropertyType element must either carry a reference to a HollowSpace object or contain a HollowSpace object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:complexContent minOccurs="0" maxOccurs="unbounded">
    <xs:sequence>
      <xs:element name="HollowSpace" type="HollowSpaceType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="IntTunnelInstallationPropertyType">
  <xs:annotation>
    <xs:documentation>Denotes the relation of a Tunnel to its internal hollow spaces. The InteriorHollowSpacePropertyType element must either carry a reference to a HollowSpace object or contain a HollowSpace object inline, but neither both nor none. </xs:documentation>
  </xs:annotation>
  <xs:complexContent minOccurs="0" maxOccurs="unbounded">
    <xs:sequence>
      <xs:element name="HollowSpace" type="HollowSpaceType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>

---

Copyright © 2012 Open Geospatial Consortium.
<xs:element name="_Opening" type="AbstractOpeningType" abstract="true" substitutionGroup="core:_CityObject"/>

<xs:element name="GenericApplicationPropertyOfOpening" type="xs:anyType" abstract="true"/>

<xs:complexType name="OpeningPropertyType">
  <xs:documentation>Denotes the relation of an _BoundarySurface to its openings (doors, windows). The OpeningPropertyType element must either carry a reference to an _Opening object or contain an _Opening object inline, but neither both nor none. </xs:documentation>
  <xs:documentation>Construction for closing an _Opening not intended for access or regress. </xs:documentation>
  <xs:documentation>Construction for closing an _Opening intended primarily for access or regress or both. </xs:documentation>
</xs:complexType>
A.12 Vegetation module

The CityGML Vegetation module is defined within the XML Schema definition file vegetation.xsd. The target namespace http://www.opengeospatial.net/citygml/vegetation/2.0 is associated with this extension module.

```xml
```

The Vegetation module is an OGC Standard. To obtain additional rights of use, visit http://www.opengeospatial.org/legal.

Type describing solitary vegetation objects, e.g., trees. Its geometry is either defined explicitly by a GML 3 geometry 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 a subclass of _CityObject, a SolitaryVegetationObject inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

```
<xs:schema targetNamespace="http://www.opengis.net/gml"
  xmlns="http://www.opengis.net/gml"
  xmlns:base="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/gml http://schemas.opengis.net/gml/2.5.1/gml.xsd">
  <xs:annotation>
    <xs:documentation>
      Copyright © 2012 Open Geospatial Consortium.
    </xs:documentation>
  </xs:annotation>
  <xs:element name="SolitaryVegetationObject" type="SolitaryVegetationObjectType" substitutionGroup="_VegetationObject" />
  <xs:element name="_GenericApplicationPropertyOfSolitaryVegetationObject" type="xs:anyType" abstract="true" />
</xs:schema>
```
A.13 WaterBody module

The CityGML WaterBody module is defined within the XML Schema definition file waterBody.xsd. The target namespace http://www.opengis.net/citygml/waterbody/2.0 is associated with this extension module.

```xml
```

CityGML is an OGC Standard. Copyright © 2012 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

The target namespace http://www.opengis.net/citygml/waterbody/2.0 associates the WaterBody module with the WaterBody.xsd file. The WaterBody module inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

The WaterBody module extends the core:AbstractCityObjectType abstract superclass for water objects. As subclass of _CityObject, a _WaterObject inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

The WaterBody module defines a WaterBodyType abstract superclass for water objects and WaterBody inherits all attributes and relations, in particular an id, names, external references, and generalization relations.

Water Bodies, e.g., lakes, rivers. As subclass of _CityObject, a WaterBody inherits all attributes and relations, in particular an id, names, external references, and generalization relations.
<xs:complexType>
  <xs:sequence minOccurs="0">  
    <xs:element ref="_WaterBoundarySurface"/>  
  </xs:sequence>

  <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
</xs:complexType>

<xsl:element name="AbstractWaterBoundarySurfaceType" abstract="true">
  <xs:annotation>
    <xs:documentation>
    A WaterBoundarySurface is a thematic object which classifies surfaces bounding a water body.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="core:AbstractCityObjectType">
        <xs:sequence>
          <xs:element name="lod4Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod3Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element name="lod2Surface" type="gml:SurfacePropertyType" minOccurs="0"/>
          <xs:element ref="_GenericApplicationPropertyOfWaterBoundarySurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xsl:element>

<xsl:element 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, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="AbstractWaterBoundarySurfaceType">
        <xs:sequence>
          <xs:element name="waterLevel" type="gml:CodeType" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xsl:element>

<xsl:element 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, and generalization relations. </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="AbstractWaterBoundarySurfaceType">
        <xs:sequence>
          <xs:element ref="_GenericApplicationPropertyOfWaterGroundSurface" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xsl:element>

<xsl:element name="BoundedByWaterSurfacePropertyType" substitutionGroup="_WaterBoundarySurface"/>

<xsl:element name="Denotes the relation of a WaterBody to its boundary surfaces, which are of type _WaterBoundarySurface. The BoundedByWaterSurfacePropertyType element must either carry a reference to a _WaterBoundarySurface object or contain a _WaterBoundarySurface object inline, but neither both nor none. </xs:documentation>
<xs:element name="_GenericApplicationPropertyOfWaterGroundSurface" type="xs:anyType" abstract="true"/>
<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, and generalization relations.</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="AbstractWaterBoundarySurfaceType">
      <xs:sequence>
        <xs:element ref="_GenericApplicationPropertyOfWaterClosureSurface" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="WaterClosureSurface" type="WaterClosureSurfaceType" substitutionGroup="_WaterBoundarySurface"/>
<xs:element name="_GenericApplicationPropertyOfWaterClosureSurface" type="xs:anyType" abstract="true"/>
</xs:schema>
A.14 TexturedSurface module [deprecated]

The CityGML TexturedSurface module is defined within the XML Schema definition file texturedSurface.xsd. The target namespace http://www.opengis.net/citygml/texturedsurface/2.0 is associated with this extension module.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/texturedsurface/2.0" xmlns:core="http://www.opengis.net/citygml/2.0"
  targetNamespace="http://www.opengis.net/citygml/texturedsurface/2.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="2.0.0">
  <xs:documentation>
  CityGML is an OGC Standard. Copyright (c) 2012 Open Geospatial Consortium.
  To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.
  </xs:documentation>
</xs:schema>
Textures can be qualified by the attribute textureType. The textureType differentiates between textures, which are specific for a certain object and are only used for that object (specific), and prototypic textures being typical for that kind of object and are used many times for all objects of that kind (typical). A typical texture may be replaced by a specific, if available. Textures may also be classified as unknown. Deprecated since CityGML version 0.4.0. Use the concepts of the CityGML Appearance module instead.

Copyright © 2012 Open Geospatial Consortium.
A.15 Schematron rules on referential integrity

The CityGML Schema repository provides Schematron rules precisely describing referential integrity constraints on CityGML property elements denoting the relation between CityGML objects. If not stated otherwise, these property elements are generally restricted to either carry a reference to a remote object using the XLink concept of GML 3.1.1 (where remote objects are located in another document or elsewhere in the same document) or contain an object inline, but neither both nor none. Within this standard document, conformance requirements on referential integrity are stated within a separate clause at the end of each chapter covering a CityGML module (e.g., see clause 10.3.9 for the CityGML Building module).

Schematron is an auxiliary constraint language. The constraints apply to context nodes within a CityGML instance document (typically an XML element) which are addressed based on XPath criteria. The rules themselves are given as XPath expressions which are evaluated for each of these nodes in order to check their validity. A CityGML instance document is considered invalid if any Schematron rule is violated. Please note, that for version 2.0 of CityGML the provided Schematron rules only limit property elements to act in either by-reference or by-value mode, but neither both nor none. Neither further restrictions on property elements nor other conformance requirements are covered. In future versions of CityGML, further Schematron rules may be added.

In CityGML, the Schematron schema used to express referential integrity rules is based on the Schematron Assertion Language version 1.5 which is also employed by GML 3.1.1. The schema is shipped as a separate file within the CityGML schema package and is accessible by the name `referentialIntegrity.sch`. It may be used to check conformance requirements on referential integrity of CityGML property elements within a CityGML instance document according to the CityGML abstract test suite provided in Annex B. However, this requires an XML validator capable of automatically processing the Schematron rules provided by the schema. Otherwise, the Schematron code can be treated merely as a formal description of the required constraints.

The following excerpt of `referentialIntegrity.sch` illustrates the abstract base rule `hrefOrContent`. This rule states XPath expressions to ensure that target objects of property elements are either given by reference or by value. It is identically specified as the corresponding rule provided by the XSD schema file `gmlBase.xsd` of GML 3.1.1. Since the `hrefOrContent` rule is abstract, it does not apply to any specific context node within a CityGML instance document but is referenced by concrete rules within the schema to ensure consistency.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
  <sch:title>Schematron constraints for CityGML 2.0</sch:title>
  <sch:ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <sch:ns prefix="bldg" uri="http://www.opengis.net/citygml/building/2.0"/>
  ...
  <sch:pattern name="Check either href or content not both">
    <sch:rule context="bldg:consistsOfBuildingPart">
      <sch:extends rule="hrefOrContent"/>
    </sch:rule>
  </sch:pattern>
  ...
</sch:schema>
```

For each CityGML property element being subject to referential integrity requirements, `referentialIntegrity.sch` contains further concrete rules based on the abstract rule `hrefOrContent`. For example, the following schema snippet applies the `hrefOrContent` rule to the property element `bldg:consistsOfBuildingPart` defined within the CityGML Building module (cf. chapter 10.3). The `bldg:consistsOfBuildingPart` element denotes the relation of a `bldg:AbstractBuilding` to its building part objects which may only be given inline or by reference.

```xml
  ...
  <sch:pattern name="hrefOrContent check on bldg:consistsOfBuildingPart">
    <sch:rule context="bldg:consistsOfBuildingPart">
      <sch:extends rule="hrefOrContent"/>
    </sch:rule>
  </sch:pattern>
  ...
```

Copyright © 2012 Open Geospatial Consortium.
Annex B
(normative)

Abstract test suite for CityGML instance documents

B.1 Test cases for mandatory conformance requirements

B.1.1 Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definitions of all employed CityGML modules. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the employed CityGML modules. |
| c) Reference | Annex A. |
| d) Test type | Basic Test. |

B.1.2 Valid CityGML profile

CityGML profile definition embedded inline the CityGML instance document

| a) Test purpose | Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions embedded inline the CityGML instance document (referenced as first approach in chapter 7.2), verify that the CityGML instance document denotes all schema definitions and corresponding XML namespaces of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile. |
| b) Test method | Inspect the instance document and check that it satisfies the rules for employing CityGML profiles described in chapter 7.2 (first approach). |
| c) Reference | Annex A, chapter 7.2 (first approach). |
| d) Test type | Basic Test. |

CityGML profile definition provided by a separate XML Schema definition file

| a) Test purpose | Verify that a profile employed by a CityGML instance document is a valid CityGML profile in accordance with the rules and guidelines stated in chapter 7.2. For CityGML profile definitions provided by a separate XML Schema definition file (referenced as second approach in chapter 7.2), verify that the profile’s XML Schema definition file itself is valid and imports all schema definitions of CityGML modules that are used to represent the data within the instance document and, thus, are part of the employed CityGML profile. The target namespace of the profile’s XML Schema definition must differ from the namespaces of the imported CityGML modules and must be given as previously unused and globally unique URI. The profile’s XML Schema definition must not contain any further content. |
| b) Test method | Validate the XML Schema definition of the CityGML profile. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the respective XML Schema specification of the CityGML profile. Inspect |
the instance document and check that it satisfies the rules for employing CityGML profiles described in chapter 7.2 (second approach).

c) Reference
Annex A, chapter 7.2 (second approach).

d) Test type
Basic Test.

### B.1.3 Conformance classes related to CityGML modules

| a) Test purpose | Verify the validity of the CityGML instance document against the conformance classes of each CityGML module that is part of the CityGML profile employed by the instance document. This may be any combination of CityGML extension modules in conjunction with the CityGML core module. |
| b) Test method | Follow the test cases provided by the conformance classes for each CityGML module in annex B.2. |
| c) Reference | Annex B.2. |
| d) Test type | Basic Test. |

### B.1.4 Spatial geometry objects

| a) Test purpose | Verify that all spatial geometry objects within a CityGML instance document adhere to the XML Schema definition of the Geography Markup Language version 3.1.1 and to the CityGML spatial model. |
| b) Test method | Inspect the instance document and check that spatial geometry objects are valid with respect to the XML Schema definition of GML version 3.1.1 and satisfy the rules of the CityGML spatial model described in chapter 8. |
| c) Reference | OGC Document No. 03-105r1, Annex A, chapter 8. |
| d) Test type | Capability Test. |

### B.1.5 Spatial topology relations

| a) Test purpose | Verify that all spatial topology relations between spatial geometry objects are expressed using the XML concept of XLinks provided by GML version 3.1.1. |
| b) Test method | Inspect the instance document and check that spatial topology relations between spatial geometry objects are valid with respect to the XLinks concept introduced by GML version 3.1.1 and satisfy the rules of the CityGML spatial model described in chapter 8. |
| c) Reference | OGC Document No. 03-105r1, Annex A, chapter 8. |
| d) Test type | Capability Test. |

### B.1.6 Address objects

| a) Test purpose | Verify that all thematic objects representing address information within a CityGML instance document adhere to the XML Schema definition of the Extensible Address Language (xAL) issued by OASIS and to the rules for representing address information in CityGML. |
| b) Test method | Inspect the instance document and check that thematic objects representing address information are valid with respect to the XML Schema definition of the OASIS Extensible Address Language and satisfy the rules for representing address information within CityGML described in chapter 10.1.4. |
| d) Test type | Capability Test. |
B.2 Conformance classes related to CityGML modules

B.2.1 CityGML Core module

Mandatory conformance requirements

<table>
<thead>
<tr>
<th>Test case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Test purpose</td>
<td>Verify that the CityGML instance document follows the CityGML Core module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents.</td>
</tr>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the CityGML Core module described in chapter 10.1 and 8.2, and especially the conformance requirements stated in chapter 10.1.6 and 8.3.3. Conformance requirements on referential integrity of CityGML property elements defined within the CityGML Core module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.1, 10.1.6, 8.2, 8.3.3, annex A.15.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Basic Test.</td>
</tr>
</tbody>
</table>

Valid CityGML instance document

<table>
<thead>
<tr>
<th>Test case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the CityGML Core module. This test case is mandatory for all CityGML instance documents.</td>
</tr>
<tr>
<td>b) Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the CityGML Core module in annex A.1. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the CityGML Core module.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Annex A.1.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Basic Test.</td>
</tr>
</tbody>
</table>

B.2.2 Appearance module

Mandatory conformance requirements

<table>
<thead>
<tr>
<th>Test case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Test purpose</td>
<td>Verify that the CityGML instance document follows the Appearance module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Appearance module. Conformance requirements on referential integrity of CityGML property elements defined within the Appearance module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</td>
</tr>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the Appearance module described in chapter 9, and especially the conformance requirements stated in chapter 9.7.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 9, 9.7, annex A.15.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

Valid CityGML instance document

<table>
<thead>
<tr>
<th>Test case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Test purpose</td>
<td>Verify the validity of the CityGML instance document against the XML Schema definition of the Appearance module. This test case is mandatory for all CityGML instance documents employing elements defined within the Appearance module.</td>
</tr>
</tbody>
</table>
b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Appearance module in annex A.2. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Appearance module.

c) Reference | Annex A.2.

d) Test type | Capability Test.

### B.2.3 Bridge module

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the Bridge module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Bridge module. Conformance requirements on referential integrity of CityGML property elements defined within the Bridge module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the Bridge module described in chapter 10.5, and especially the conformance requirements stated in chapter 10.5.8. |
| c) Reference | Chapter 10.5, 10.5.8, annex A.15. |
| d) Test type | Capability Test. |

**Valid CityGML instance document**

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Bridge module. This test case is mandatory for all CityGML instance documents employing elements defined within the Bridge module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Bridge module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Bridge module. |
| c) Reference | Annex A.3. |
| d) Test type | Capability Test. |

### B.2.4 Building module

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the Building module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Building module. Conformance requirements on referential integrity of CityGML property elements defined within the Building module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the Building module described in chapter 10.3, and especially the conformance requirements stated in chapter 10.3.9. |
| c) Reference | Chapter 10.3, 10.3.9, annex A.15. |
| d) Test type | Capability Test. |

**Valid CityGML instance document**

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Building module. This test case is mandatory for all CityGML instance documents employing elements defined within the Building module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Building module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Building module. |
| d) Test type | Capability Test. |

**B.2.5 CityFurniture module**

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the CityFurniture module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the CityFurniture module. Conformance requirements on referential integrity of CityGML property elements defined within the CityFurniture module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the CityFurniture module described in chapter 10.9, and especially the conformance requirements stated in chapter 10.9.4. |
| c) Reference | Chapter 10.9, 10.9.4, annex A.15. |
| d) Test type | Capability Test. |

**Valid CityGML instance document**

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the CityFurniture module. This test case is mandatory for all CityGML instance documents employing elements defined within the CityFurniture module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the CityFurniture module in annex A.5. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the CityFurniture module. |
| c) Reference | Annex A.5. |
| d) Test type | Capability Test. |

**B.2.6 CityObjectGroup module**

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the CityObjectGroup module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This |
### Test case

The test case is mandatory for all CityGML instance documents employing elements defined within the `CityObjectGroup` module. Conformance requirements on referential integrity of CityGML property elements defined within the `CityObjectGroup` module may be additionally validated using the constraints provided by the Schematron schema `referentialIntegrity.sch` in accordance with the rules and guidelines stated in annex A.15.

#### b) Test method

Inspect the instance document and check that it satisfies the rules of the `CityObjectGroup` module described in chapter 10.11, and especially the conformance requirements stated in chapter 10.11.2.

#### c) Reference

Chapter 10.11, 10.11.2, annex A.15.

#### d) Test type

Capability Test.

---

### Valid CityGML instance document

#### a) Test purpose

Verify the validity of the CityGML instance document against the XML Schema definition of the `CityObjectGroup` module. This test case is mandatory for all CityGML instance documents employing elements defined within the `CityObjectGroup` module.

#### b) Test method

Validate the CityGML XML instance document against the XML Schema definition of the `CityObjectGroup` module in annex A.6. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the `CityObjectGroup` module.

#### c) Reference


#### d) Test type

Capability Test.

---

### B.2.7 Generics module

#### Mandatory conformance requirements

#### a) Test purpose

Verify that the CityGML instance document follows the `Generics` module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the `Generics` module. Conformance requirements on referential integrity of CityGML property elements defined within the `Generics` module may be additionally validated using the constraints provided by the Schematron schema `referentialIntegrity.sch` in accordance with the rules and guidelines stated in annex A.15.

#### b) Test method

Inspect the instance document and check that it satisfies the rules of the `Generics` module described in chapter 10.12, and especially the conformance requirements stated in chapter 10.12.2.

#### c) Reference


#### d) Test type

Capability Test.

---

### Valid CityGML instance document

#### a) Test purpose

Verify the validity of the CityGML instance document against the XML Schema definition of the `Generics` module. This test case is mandatory for all CityGML instance documents employing elements defined within the `Generics` module.

#### b) Test method

Validate the CityGML XML instance document against the XML Schema definition of the `Generics` module in annex A.7. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the `Generics` module.

#### c) Reference

Annex A.7.
**B.2.8 LandUse module**

**Mandatory conformance requirements**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the LandUse module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the LandUse module. Conformance requirements on referential integrity of CityGML property elements defined within the LandUse module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the LandUse module described in chapter 10.10, and especially the conformance requirements stated in chapter 10.10.3.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.10, 10.10.3, annex A.15.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

**Valid CityGML instance document**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify the validity of the CityGML instance document against the XML Schema definition of the LandUse module. This test case is mandatory for all CityGML instance documents employing elements defined within the LandUse module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the LandUse module in annex A.8. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the LandUse module.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Annex A.8.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

**B.2.9 Relief module**

**Mandatory conformance requirements**

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the Relief module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Relief module. Conformance requirements on referential integrity of CityGML property elements defined within the Relief module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the Relief module described in chapter 10.2, and especially the conformance requirements stated in chapter 10.2.6.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.2, 10.2.6, annex A.15.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>
Valid CityGML instance document

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify the validity of the CityGML instance document against the XML Schema definition of the Relief module. This test case is mandatory for all CityGML instance documents employing elements defined within the Relief module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the Relief module in annex A.9. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Relief module.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

B.2.10 Transportation module

Mandatory conformance requirements

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the Transportation module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Transportation module. Conformance requirements on referential integrity of CityGML property elements defined within the Transportation module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Inspect the instance document and check that it satisfies the rules of the Transportation module described in chapter 10.7, and especially the conformance requirements stated in chapter 10.7.5.</td>
</tr>
<tr>
<td>c) Reference</td>
<td>Chapter 10.7, 10.7.5, annex A.15.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

Valid CityGML instance document

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify the validity of the CityGML instance document against the XML Schema definition of the Transportation module. This test case is mandatory for all CityGML instance documents employing elements defined within the Transportation module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b) Test method</td>
<td>Validate the CityGML XML instance document against the XML Schema definition of the Transportation module in annex A.10. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Transportation module.</td>
</tr>
<tr>
<td>d) Test type</td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>

B.2.11 Tunnel module

Mandatory conformance requirements

<table>
<thead>
<tr>
<th>a) Test purpose</th>
<th>Verify that the CityGML instance document follows the Tunnel module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Tunnel module. Conformance requirements on referential integrity of CityGML property elements defined within the Tunnel module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15.</th>
</tr>
</thead>
</table>
b) Test method | Inspect the instance document and check that it satisfies the rules of the Tunnel module described in chapter 10.4, and especially the conformance requirements stated in chapter 10.4.8.
---|---
c) Reference | Chapter 10.4, 10.4.8, annex A.15.
d) Test type | Capability Test.

Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Tunnel module. This test case is mandatory for all CityGML instance documents employing elements defined within the Tunnel module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Tunnel module in annex A.3. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Tunnel module. |
| c) Reference | Annex A.11. |
| d) Test type | Capability Test. |

B.2.12 Vegetation module

Mandatory conformance requirements

| a) Test purpose | Verify that the CityGML instance document follows the Vegetation module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the Vegetation module. Conformance requirements on referential integrity of CityGML property elements defined within the Vegetation module may be additionally validated using the constraints provided by the Schematron schema referentialIntegrity.sch in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the Vegetation module described in chapter 10.8, and especially the conformance requirements stated in chapter 10.8.6. |
| c) Reference | Chapter 10.8, 10.8.6, annex A.15. |
| d) Test type | Capability Test. |

Valid CityGML instance document

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the Vegetation module. This test case is mandatory for all CityGML instance documents employing elements defined within the Vegetation module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the Vegetation module in annex A.12. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the Vegetation module. |
| c) Reference | Annex A.12. |
| d) Test type | Capability Test. |
### B.2.13 WaterBody module

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the *WaterBody* module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *WaterBody* module. Conformance requirements on referential integrity of CityGML property elements defined within the *WaterBody* module may be additionally validated using the constraints provided by the Schematron schema `referentialIntegrity.sch` in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the *WaterBody* module described in chapter 10.6, and especially the conformance requirements stated in chapter 10.6.4. |
| c) Reference | Chapter 10.6, 10.6.4, annex A.15. |
| d) Test type | Capability Test. |

**Valid CityGML instance document**

| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the *WaterBody* module. This test case is mandatory for all CityGML instance documents employing elements defined within the *WaterBody* module. |
| b) Test method | Validate the CityGML XML instance document against the XML Schema definition of the *WaterBody* module in annex A.13. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the *WaterBody* module. |
| d) Test type | Capability Test. |

### B.2.14 TexturedSurface module [deprecated]

**Mandatory conformance requirements**

| a) Test purpose | Verify that the CityGML instance document follows the *TexturedSurface* module’s rules for encoding of objects and properties and adheres to all its conformance requirements. This test case is mandatory for all CityGML instance documents employing elements defined within the *TexturedSurface* module. Conformance requirements on referential integrity of CityGML property elements defined within the *TexturedSurface* module may be additionally validated using the constraints provided by the Schematron schema `referentialIntegrity.sch` in accordance with the rules and guidelines stated in annex A.15. |
| b) Test method | Inspect the instance document and check that it satisfies the rules of the *TexturedSurface* module described in chapter 9.8, and especially the conformance requirements stated in chapter 9.8.2. |
| d) Test type | Capability Test. |

**Valid CityGML instance documents**

<p>| a) Test purpose | Verify the validity of the CityGML instance document against the XML Schema definition of the <em>TexturedSurface</em> module. This test case is mandatory for all CityGML instance documents employing elements defined within the <em>TexturedSurface</em> module. |</p>
<table>
<thead>
<tr>
<th><strong>b) Test method</strong></th>
<th>Validate the CityGML XML instance document against the XML Schema definition of the <em>TexturedSurface</em> module in annex A.14. The process may be using an appropriate software tool for validation or be a manual process that checks all relevant definitions from the <em>TexturedSurface</em> module.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>c) Reference</strong></td>
<td>Annex A.14.</td>
</tr>
<tr>
<td><strong>d) Test type</strong></td>
<td>Capability Test.</td>
</tr>
</tbody>
</table>
Annex C
(informative)

Code lists proposed by the SIG 3D

This annex exemplifies the specification of code lists for enumerative attributes of type `gml:CodeType` and provides proposals for selected attributes. Please note that this annex is non-normative and the presented code lists are neither mandatory nor complete. The governance of code lists is explicitly decoupled from the governance of the CityGML schema and specification (cf. chapter 10.14). Thus, code lists can be freely defined outside the CityGML specification by any organization or information community according to their information needs, and there shall be one authority per code list who is in charge of maintenance and updating the contents.

The code lists presented in this annex are managed and maintained by the SIG 3D and are accessible at [http://www.sig3d.de/codelists/standard/](http://www.sig3d.de/codelists/standard/). They completely comprise the contents of the non-normative code lists given in the previous CityGML 1.0 specification document (cf. Annex C of OGC Doc. No. 08-007r1). In order to maintain backwards compatibility, none of the previous entries has been modified or changed in meaning. Additional code lists have been added for some of the newly introduced enumerative attributes in CityGML 2.0, for example in the context of the Bridge and Tunnel module. The SIG 3D code lists will be complemented subsequently. The previous version 1.0 code lists are still accessible from the OGC repository at [http://schemas.opengis.net/citygml/codelists/](http://schemas.opengis.net/citygml/codelists/).

The SIG 3D code lists are implemented as *simple dictionaries* following the GML 3.1.1 Simple Dictionary Profile and can be directly referenced in CityGML instance documents. For this purpose, the `codeSpace` attribute declared by `gml:CodeType` has to point to the code list of the corresponding enumerative feature attribute. The following naming pattern is applied by the SIG 3D in order to derive a globally unique URL in the `http` schema for each code list:

```
http://www.sig3d.org/codelists/standard/
  <moduleName>/<version>/<FeatureTypeName>_<propertyName>.xml
```

The placeholder `<moduleName>` has to be replaced with the name of the CityGML module in which the feature and the attribute are defined. The module name has to be given in small letters (e.g., `building` or `cityfurniture`). The `<version>` field defines the CityGML version for which the code list is specified. Finally, `<FeatureTypeName>` denotes the name of the feature type for which the attribute is modeled and `<propertyName>` is to be replaced with the attribute name itself. For both placeholders, the XML element name given in the CityGML schema document where the feature is specified has to be chosen.

For example, the `codeSpace` value identifying the SIG 3D code list for the enumerative attribute `roofType` of the feature _AbstractBuilding_ (Building module, cf. chapter 10.3) is:

```
http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml
```

The following excerpt demonstrates how to reference the SIG 3D code list for `roofType` from within a CityGML instance document. The attribute value 1000 denotes a `flat roof` according to this code list (cf. subclause C.1).

```
<building:Building>
  <building:roofType
    codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1000</building:roofType>
...;
</building:Building>
```

The SIG 3D code lists can be downloaded from these URLs in order to allow applications to automatically validate the coded attribute values.

The entire code list for building roof types is given as example below, while the others are depicted in the following subclauses in tabular form for space reasons. The subclauses are arranged according to thematic modules.
Code list for roof types  (http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml)

```xml
    xsi:schemaLocation="http://www.opengis.net/gml
http://schemas.opengis.net/gml/3.1.1/profiles/SimpleDictionary/1.0.0/gmlSimpleDictionaryProfile.xsd"
    gml:id="_AbstractBuilding_roofType">
    <gml:name>_AbstractBuilding_roofType</gml:name>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id1"/>
        <gml:description>flat roof</gml:description>
        <gml:name>1000</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id2"/>
        <gml:description>monopitch roof</gml:description>
        <gml:name>1010</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id3"/>
        <gml:description>skirt pent roof</gml:description>
        <gml:name>1020</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id4"/>
        <gml:description>gabled roof</gml:description>
        <gml:name>1030</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id5"/>
        <gml:description>hipped roof</gml:description>
        <gml:name>1040</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id6"/>
        <gml:description>half-hipped roof</gml:description>
        <gml:name>1050</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id7"/>
        <gml:description>mansard roof</gml:description>
        <gml:name>1060</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id8"/>
        <gml:description>pavilion roof</gml:description>
        <gml:name>1070</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id9"/>
        <gml:description>cone roof</gml:description>
        <gml:name>1080</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id10"/>
        <gml:description>copula roof</gml:description>
        <gml:name>1090</gml:name>
    </gml:dictionaryEntry>
    <gml:dictionaryEntry>
        <gml:Definition gml:id="id11"/>
        <gml:description>shed roof</gml:description>
        <gml:name>1100</gml:name>
    </gml:dictionaryEntry>
</gml:Dictionary>
```
<gml:dictionary>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id12">
      <gml:description>arch roof</gml:description>
      <gml:name>1110</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id13">
      <gml:description>pyramidal broach roof</gml:description>
      <gml:name>1120</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
  <gml:dictionaryEntry>
    <gml:Definition gml:id="id14">
      <gml:description>combination of roof forms</gml:description>
      <gml:name>1130</gml:name>
    </gml:Definition>
  </gml:dictionaryEntry>
</gml:Dictionary>
## C.1 Building module

### Code list of the AbstractBuilding attribute class

- [Link to Code list](http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_class.xml)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>habitation</td>
<td>1100</td>
</tr>
<tr>
<td>1010</td>
<td>sanitation</td>
<td>1110</td>
</tr>
<tr>
<td>1020</td>
<td>administration</td>
<td>1120</td>
</tr>
<tr>
<td>1030</td>
<td>business, trade</td>
<td>1130</td>
</tr>
<tr>
<td>1040</td>
<td>catering</td>
<td>1140</td>
</tr>
<tr>
<td>1050</td>
<td>recreation</td>
<td>1150</td>
</tr>
<tr>
<td>1060</td>
<td>sport</td>
<td>1160</td>
</tr>
<tr>
<td>1070</td>
<td>culture</td>
<td>1170</td>
</tr>
<tr>
<td>1080</td>
<td>church institution</td>
<td>1180</td>
</tr>
<tr>
<td>1090</td>
<td>agriculture, forestry</td>
<td></td>
</tr>
</tbody>
</table>

### Code list of the AbstractBuilding attributes function and usage

- [Link to Code list](http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml)
- [Link to Code list](http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_usage.xml)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>residential building</td>
<td>1840</td>
</tr>
<tr>
<td>1010</td>
<td>tenement</td>
<td>1850</td>
</tr>
<tr>
<td>1020</td>
<td>hostel</td>
<td>1860</td>
</tr>
<tr>
<td>1030</td>
<td>residential- and administration building</td>
<td>1870</td>
</tr>
<tr>
<td>1040</td>
<td>residential- and office building</td>
<td>1880</td>
</tr>
<tr>
<td>1050</td>
<td>residential- and business building</td>
<td>1890</td>
</tr>
<tr>
<td>1060</td>
<td>residential- and plant building</td>
<td>1900</td>
</tr>
<tr>
<td>1070</td>
<td>agrarian- and forestry building</td>
<td>1910</td>
</tr>
<tr>
<td>1080</td>
<td>residential- and commercial building</td>
<td>1920</td>
</tr>
<tr>
<td>1090</td>
<td>forester's lodge</td>
<td>1930</td>
</tr>
<tr>
<td>1100</td>
<td>holiday house</td>
<td>1940</td>
</tr>
<tr>
<td>1110</td>
<td>summer house</td>
<td>1950</td>
</tr>
<tr>
<td>1120</td>
<td>office building</td>
<td>1960</td>
</tr>
<tr>
<td>1130</td>
<td>credit institution</td>
<td>1970</td>
</tr>
<tr>
<td>1140</td>
<td>insurance</td>
<td>1980</td>
</tr>
<tr>
<td>1150</td>
<td>business building</td>
<td>1990</td>
</tr>
<tr>
<td>1160</td>
<td>department store</td>
<td>2000</td>
</tr>
<tr>
<td>1170</td>
<td>shopping centre</td>
<td>2010</td>
</tr>
<tr>
<td>1180</td>
<td>kiosk</td>
<td>2020</td>
</tr>
<tr>
<td>1190</td>
<td>pharmacy</td>
<td>2030</td>
</tr>
<tr>
<td>1200</td>
<td>pavilion</td>
<td>2040</td>
</tr>
<tr>
<td>1210</td>
<td>hotel</td>
<td>2050</td>
</tr>
<tr>
<td>1220</td>
<td>youth hostel</td>
<td>2060</td>
</tr>
<tr>
<td>1230</td>
<td>campsite building</td>
<td>2070</td>
</tr>
<tr>
<td>1240</td>
<td>restaurant</td>
<td>2080</td>
</tr>
<tr>
<td>1250</td>
<td>cantine</td>
<td>2090</td>
</tr>
<tr>
<td>1260</td>
<td>recreational site</td>
<td>2100</td>
</tr>
<tr>
<td>1270</td>
<td>function room</td>
<td>2110</td>
</tr>
<tr>
<td>1280</td>
<td>cinema</td>
<td>2120</td>
</tr>
<tr>
<td>1290</td>
<td>bowling alley</td>
<td>2130</td>
</tr>
<tr>
<td>1300</td>
<td>casino</td>
<td>2140</td>
</tr>
<tr>
<td>1310</td>
<td>industrial building</td>
<td>2150</td>
</tr>
<tr>
<td>1320</td>
<td>factory</td>
<td>2160</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Code</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1330</td>
<td>workshop</td>
<td>2170</td>
</tr>
<tr>
<td>1340</td>
<td>petrol / gas station</td>
<td>2180</td>
</tr>
<tr>
<td>1350</td>
<td>washing plant</td>
<td>2190</td>
</tr>
<tr>
<td>1360</td>
<td>cold store</td>
<td>2200</td>
</tr>
<tr>
<td>1370</td>
<td>depot</td>
<td>2210</td>
</tr>
<tr>
<td>1380</td>
<td>building for research purposes</td>
<td>2220</td>
</tr>
<tr>
<td>1390</td>
<td>quarry</td>
<td>2230</td>
</tr>
<tr>
<td>1400</td>
<td>salt works</td>
<td>2240</td>
</tr>
<tr>
<td>1410</td>
<td>miscellaneous industrial building</td>
<td>2250</td>
</tr>
<tr>
<td>1420</td>
<td>mill</td>
<td>2260</td>
</tr>
<tr>
<td>1430</td>
<td>windmill</td>
<td>2270</td>
</tr>
<tr>
<td>1440</td>
<td>water mill</td>
<td>2280</td>
</tr>
<tr>
<td>1450</td>
<td>bucket elevator</td>
<td>2290</td>
</tr>
<tr>
<td>1460</td>
<td>weather station</td>
<td>2300</td>
</tr>
<tr>
<td>1470</td>
<td>traffic assets office</td>
<td>2310</td>
</tr>
<tr>
<td>1480</td>
<td>street maintenance</td>
<td>2320</td>
</tr>
<tr>
<td>1490</td>
<td>waiting hall</td>
<td>2330</td>
</tr>
<tr>
<td>1500</td>
<td>signal control box</td>
<td>2340</td>
</tr>
<tr>
<td>1510</td>
<td>engine shed</td>
<td>2350</td>
</tr>
<tr>
<td>1520</td>
<td>signal box or stop signal</td>
<td>2360</td>
</tr>
<tr>
<td>1530</td>
<td>plant building for air traffic</td>
<td>2370</td>
</tr>
<tr>
<td>1540</td>
<td>hangar</td>
<td>2380</td>
</tr>
<tr>
<td>1550</td>
<td>plant building for shipping</td>
<td>2390</td>
</tr>
<tr>
<td>1560</td>
<td>shipyard</td>
<td>2400</td>
</tr>
<tr>
<td>1570</td>
<td>dock</td>
<td>2410</td>
</tr>
<tr>
<td>1580</td>
<td>plant building for canal lock</td>
<td>2420</td>
</tr>
<tr>
<td>1590</td>
<td>boathouse</td>
<td>2430</td>
</tr>
<tr>
<td>1600</td>
<td>plant building for cablecar</td>
<td>2440</td>
</tr>
<tr>
<td>1610</td>
<td>multi-storey car park</td>
<td>2450</td>
</tr>
<tr>
<td>1620</td>
<td>parking level</td>
<td>2460</td>
</tr>
<tr>
<td>1630</td>
<td>garage</td>
<td>2470</td>
</tr>
<tr>
<td>1640</td>
<td>vehicle hall</td>
<td>2480</td>
</tr>
<tr>
<td>1650</td>
<td>underground garage</td>
<td>2490</td>
</tr>
<tr>
<td>1660</td>
<td>building for supply</td>
<td>2500</td>
</tr>
<tr>
<td>1670</td>
<td>waterworks</td>
<td>2510</td>
</tr>
<tr>
<td>1680</td>
<td>pump station</td>
<td>2520</td>
</tr>
<tr>
<td>1690</td>
<td>water basin</td>
<td>2530</td>
</tr>
<tr>
<td>1700</td>
<td>electric power station</td>
<td>2540</td>
</tr>
<tr>
<td>1710</td>
<td>transformer station</td>
<td>2550</td>
</tr>
<tr>
<td>1720</td>
<td>converter</td>
<td>2560</td>
</tr>
<tr>
<td>1730</td>
<td>reactor</td>
<td>2570</td>
</tr>
<tr>
<td>1740</td>
<td>turbine house</td>
<td>2580</td>
</tr>
<tr>
<td>1750</td>
<td>boiler house</td>
<td>2590</td>
</tr>
<tr>
<td>1760</td>
<td>building for telecommunications</td>
<td>2600</td>
</tr>
<tr>
<td>1770</td>
<td>gas works</td>
<td>2610</td>
</tr>
<tr>
<td>1780</td>
<td>heat plant</td>
<td>2620</td>
</tr>
<tr>
<td>1790</td>
<td>pumping station</td>
<td>2630</td>
</tr>
<tr>
<td>1800</td>
<td>building for disposal</td>
<td>2640</td>
</tr>
<tr>
<td>1810</td>
<td>building for effluent disposal</td>
<td>2650</td>
</tr>
<tr>
<td>1820</td>
<td>building for filter plant</td>
<td>2700</td>
</tr>
<tr>
<td>1830</td>
<td>toilet</td>
<td></td>
</tr>
</tbody>
</table>
### Code list of the BuildingFurniture attribute class

**http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_class.xml**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>habituation</td>
</tr>
<tr>
<td>1010</td>
<td>sanitation</td>
</tr>
<tr>
<td>1020</td>
<td>administration</td>
</tr>
<tr>
<td>1030</td>
<td>business, trade</td>
</tr>
<tr>
<td>1040</td>
<td>catering</td>
</tr>
<tr>
<td>1050</td>
<td>recreation</td>
</tr>
<tr>
<td>1060</td>
<td>sport</td>
</tr>
<tr>
<td>1070</td>
<td>culture</td>
</tr>
<tr>
<td>1080</td>
<td>church institution</td>
</tr>
<tr>
<td>1090</td>
<td>agriculture, forestry</td>
</tr>
<tr>
<td>1100</td>
<td>schools, education, research</td>
</tr>
<tr>
<td>1110</td>
<td>maintenance, waste management</td>
</tr>
<tr>
<td>1120</td>
<td>healthcare</td>
</tr>
<tr>
<td>1130</td>
<td>communicating</td>
</tr>
<tr>
<td>1140</td>
<td>security</td>
</tr>
<tr>
<td>1150</td>
<td>storage</td>
</tr>
<tr>
<td>1160</td>
<td>industry</td>
</tr>
<tr>
<td>1170</td>
<td>traffic</td>
</tr>
<tr>
<td>1180</td>
<td>function</td>
</tr>
</tbody>
</table>

### Code list of the BuildingFurniture attributes function and usage

**http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_function.xml**

**http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_usage.xml**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>cupboard</td>
</tr>
<tr>
<td>1010</td>
<td>wardrobe</td>
</tr>
<tr>
<td>1020</td>
<td>cabinet</td>
</tr>
<tr>
<td>1030</td>
<td>sideboard</td>
</tr>
<tr>
<td>1040</td>
<td>locker</td>
</tr>
<tr>
<td>1050</td>
<td>tool cabinet</td>
</tr>
<tr>
<td>1060</td>
<td>shelf</td>
</tr>
<tr>
<td>1100</td>
<td>rack</td>
</tr>
<tr>
<td>1110</td>
<td>coat stand</td>
</tr>
<tr>
<td>1120</td>
<td>table</td>
</tr>
<tr>
<td>1130</td>
<td>dining table</td>
</tr>
<tr>
<td>1140</td>
<td>coffee table</td>
</tr>
<tr>
<td>1150</td>
<td>desk</td>
</tr>
<tr>
<td>1160</td>
<td>bedside cabinet</td>
</tr>
<tr>
<td>1170</td>
<td>baby changing table</td>
</tr>
<tr>
<td>1180</td>
<td>bar</td>
</tr>
<tr>
<td>1190</td>
<td>pool table</td>
</tr>
<tr>
<td>1200</td>
<td>snooker table</td>
</tr>
<tr>
<td>1210</td>
<td>roulette table</td>
</tr>
<tr>
<td>1220</td>
<td>work bench</td>
</tr>
<tr>
<td>1230</td>
<td>chair</td>
</tr>
<tr>
<td>1240</td>
<td>bench</td>
</tr>
<tr>
<td>1250</td>
<td>office chair</td>
</tr>
<tr>
<td>1260</td>
<td>sofa</td>
</tr>
<tr>
<td>1270</td>
<td>rocking chair</td>
</tr>
<tr>
<td>1280</td>
<td>bar stool</td>
</tr>
<tr>
<td>1290</td>
<td>armchair</td>
</tr>
<tr>
<td>1300</td>
<td>bed</td>
</tr>
<tr>
<td>1310</td>
<td>crib</td>
</tr>
<tr>
<td>1320</td>
<td>bunk bed</td>
</tr>
<tr>
<td>1330</td>
<td>cradle</td>
</tr>
<tr>
<td>1340</td>
<td>cot</td>
</tr>
<tr>
<td>1350</td>
<td>stretcher</td>
</tr>
<tr>
<td>1360</td>
<td>standard lamp</td>
</tr>
<tr>
<td>1370</td>
<td>ceiling light</td>
</tr>
<tr>
<td>1380</td>
<td>lighting</td>
</tr>
<tr>
<td>1390</td>
<td>oil tank</td>
</tr>
<tr>
<td>1400</td>
<td>water tank</td>
</tr>
<tr>
<td>1410</td>
<td>gas tank</td>
</tr>
<tr>
<td>1420</td>
<td>fuel tank</td>
</tr>
<tr>
<td>1430</td>
<td>milk tank</td>
</tr>
<tr>
<td>1440</td>
<td>steel tank</td>
</tr>
<tr>
<td>1450</td>
<td>fire protection appliance</td>
</tr>
<tr>
<td>1500</td>
<td>fire extinguishing system</td>
</tr>
<tr>
<td>1510</td>
<td>fire alarm</td>
</tr>
</tbody>
</table>

Copyright © 2012 Open Geospatial Consortium.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1530</td>
<td>spotlight</td>
<td>2533</td>
<td>fire extinguisher</td>
</tr>
<tr>
<td>1600</td>
<td>electric appliances</td>
<td>2540</td>
<td>switch board</td>
</tr>
<tr>
<td>1610</td>
<td>television set</td>
<td>2550</td>
<td>lifting platform</td>
</tr>
<tr>
<td>1620</td>
<td>video recorder</td>
<td>2560</td>
<td>compressed air system</td>
</tr>
<tr>
<td>1630</td>
<td>stereo unit</td>
<td>2570</td>
<td>loud-speaker</td>
</tr>
<tr>
<td>1700</td>
<td>kitchen appliances</td>
<td>2580</td>
<td>microphone</td>
</tr>
<tr>
<td>1710</td>
<td>cooker</td>
<td>2600</td>
<td>sports equipment</td>
</tr>
<tr>
<td>1720</td>
<td>oven</td>
<td>2610</td>
<td>goal posts</td>
</tr>
<tr>
<td>1730</td>
<td>refrigerator</td>
<td>2620</td>
<td>basketball basket</td>
</tr>
<tr>
<td>1740</td>
<td>coffee machine</td>
<td>2630</td>
<td>volleyball net</td>
</tr>
<tr>
<td>1750</td>
<td>toaster</td>
<td>2640</td>
<td>gymnastic apparatus</td>
</tr>
<tr>
<td>1760</td>
<td>kettle</td>
<td>2650</td>
<td>diving platform</td>
</tr>
<tr>
<td>1770</td>
<td>microwave</td>
<td>2660</td>
<td>swimming pool</td>
</tr>
<tr>
<td>1780</td>
<td>dish washer</td>
<td>2700</td>
<td>sales promotion furniture</td>
</tr>
<tr>
<td>1800</td>
<td>laundry equipment</td>
<td>2710</td>
<td>display panel</td>
</tr>
<tr>
<td>1810</td>
<td>washing machine</td>
<td>2720</td>
<td>billboard</td>
</tr>
<tr>
<td>1820</td>
<td>ironing machine</td>
<td>2730</td>
<td>display cabinet</td>
</tr>
<tr>
<td>1830</td>
<td>rotary iron (mangle)</td>
<td>2800</td>
<td>functional furniture</td>
</tr>
<tr>
<td>1840</td>
<td>laundry tumble drier</td>
<td>2805</td>
<td>ashtray</td>
</tr>
<tr>
<td>1850</td>
<td>spin drier</td>
<td>2810</td>
<td>lectern</td>
</tr>
<tr>
<td>1900</td>
<td>technical office equipment</td>
<td>2815</td>
<td>stage</td>
</tr>
<tr>
<td>1910</td>
<td>copy machine</td>
<td>2820</td>
<td>blackboard</td>
</tr>
<tr>
<td>1920</td>
<td>scanner</td>
<td>2825</td>
<td>screen</td>
</tr>
<tr>
<td>1930</td>
<td>plotter</td>
<td>2830</td>
<td>mapstand</td>
</tr>
<tr>
<td>1940</td>
<td>printer</td>
<td>2835</td>
<td>rubbish bin</td>
</tr>
<tr>
<td>1950</td>
<td>screen</td>
<td>2840</td>
<td>sauna</td>
</tr>
<tr>
<td>1960</td>
<td>computer</td>
<td>2845</td>
<td>carpet</td>
</tr>
<tr>
<td>1970</td>
<td>overhead projector</td>
<td>2850</td>
<td>wall clock</td>
</tr>
<tr>
<td>1980</td>
<td>video projector</td>
<td>2855</td>
<td>curtain</td>
</tr>
<tr>
<td>2000</td>
<td>sanitation equipment</td>
<td>2860</td>
<td>mirror</td>
</tr>
</tbody>
</table>

**Code list of the BuildingInstallation attribute class**

http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_class.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>outer characteristics</td>
</tr>
<tr>
<td>1010</td>
<td>inner characteristics</td>
</tr>
<tr>
<td>1020</td>
<td>waste management</td>
</tr>
<tr>
<td>1030</td>
<td>maintenance</td>
</tr>
<tr>
<td>1040</td>
<td>communicating</td>
</tr>
<tr>
<td>1050</td>
<td>security</td>
</tr>
<tr>
<td>1060</td>
<td>others</td>
</tr>
</tbody>
</table>

**Code list of the BuildingInstallation attributes function and usage**

http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_function.xml
http://www.sig3d.org/codelists/standard/building/2.0/BuildingInstallation_usage.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>balcony</td>
</tr>
<tr>
<td>1010</td>
<td>winter garden</td>
</tr>
<tr>
<td>1020</td>
<td>arcade</td>
</tr>
<tr>
<td>1030</td>
<td>chimney (part of a building)</td>
</tr>
<tr>
<td>1040</td>
<td>tower (part of a building)</td>
</tr>
<tr>
<td>1050</td>
<td>column</td>
</tr>
<tr>
<td>1060</td>
<td>stairs</td>
</tr>
<tr>
<td>1070</td>
<td>others</td>
</tr>
</tbody>
</table>

**Code list of the IntBuildingInstallation attribute class**

http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_class.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Heating, Ventilation, Climate</td>
</tr>
<tr>
<td>2000</td>
<td>Safety</td>
</tr>
<tr>
<td>3000</td>
<td>Illumination</td>
</tr>
<tr>
<td>6000</td>
<td>Statics</td>
</tr>
<tr>
<td>7000</td>
<td>Entertainment</td>
</tr>
<tr>
<td>8000</td>
<td>Miscellaneous</td>
</tr>
<tr>
<td>Code list of the <code>IntBuildingInstallation</code> attributes function and usage</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_function.xml">http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_function.xml</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_usage.xml">http://www.sig3d.org/codelists/standard/building/2.0/IntBuildingInstallation_usage.xml</a></td>
<td></td>
</tr>
<tr>
<td>1010 Radiator</td>
<td>3020 Light switch</td>
</tr>
<tr>
<td>1020 Oven</td>
<td>5030 Power point</td>
</tr>
<tr>
<td>1030 Fireside</td>
<td>5020 Cable</td>
</tr>
<tr>
<td>1040 Ventilator</td>
<td>7010 Rafter</td>
</tr>
<tr>
<td>1050 Air Conditioning</td>
<td>7020 Column</td>
</tr>
<tr>
<td>5010 Pipe</td>
<td>8010 Railing</td>
</tr>
<tr>
<td>3010 Lamp</td>
<td>8020 Stair</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the <code>_AbstractBuilding</code> attribute roofType</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml</a></td>
</tr>
<tr>
<td>1000 flat roof</td>
</tr>
<tr>
<td>1010 monopitch roof</td>
</tr>
<tr>
<td>1020 dual pent roof</td>
</tr>
<tr>
<td>1030 gabled roof</td>
</tr>
<tr>
<td>1040 hipped roof</td>
</tr>
<tr>
<td>1050 half-hipped roof</td>
</tr>
<tr>
<td>1060 mansard roof</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the <code>Room</code> attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/Room_class.xml">http://www.sig3d.org/codelists/standard/building/2.0/Room_class.xml</a></td>
</tr>
<tr>
<td>1000 habitation</td>
</tr>
<tr>
<td>1010 administration</td>
</tr>
<tr>
<td>1020 business, trade</td>
</tr>
<tr>
<td>1030 catering</td>
</tr>
<tr>
<td>1040 recreation</td>
</tr>
<tr>
<td>1050 church institution</td>
</tr>
<tr>
<td>1060 agriculture, forestry</td>
</tr>
<tr>
<td>1070 schools, education, research</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the <code>Room</code> attributes function and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/Room_function.xml">http://www.sig3d.org/codelists/standard/building/2.0/Room_function.xml</a></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/building/2.0/Room_usage.xml">http://www.sig3d.org/codelists/standard/building/2.0/Room_usage.xml</a></td>
</tr>
<tr>
<td>1000 living room</td>
</tr>
<tr>
<td>1010 bedroom</td>
</tr>
<tr>
<td>1020 kitchen</td>
</tr>
<tr>
<td>1030 hall</td>
</tr>
<tr>
<td>1040 bath, washroom</td>
</tr>
<tr>
<td>1050 toilet</td>
</tr>
<tr>
<td>1060 stairs</td>
</tr>
<tr>
<td>1070 home office</td>
</tr>
<tr>
<td>1080 utility room</td>
</tr>
<tr>
<td>1090 dining room</td>
</tr>
<tr>
<td>1100 common room</td>
</tr>
<tr>
<td>1110 party room</td>
</tr>
<tr>
<td>1120 nursery</td>
</tr>
<tr>
<td>1130</td>
</tr>
<tr>
<td>1140</td>
</tr>
<tr>
<td>1150</td>
</tr>
<tr>
<td>1160</td>
</tr>
<tr>
<td>1170</td>
</tr>
<tr>
<td>1180</td>
</tr>
<tr>
<td>1190</td>
</tr>
<tr>
<td>1200</td>
</tr>
<tr>
<td>1210</td>
</tr>
<tr>
<td>1220</td>
</tr>
<tr>
<td>1300</td>
</tr>
<tr>
<td>1310</td>
</tr>
<tr>
<td>1320</td>
</tr>
<tr>
<td>1330</td>
</tr>
<tr>
<td>1340</td>
</tr>
<tr>
<td>1350</td>
</tr>
<tr>
<td>1360</td>
</tr>
<tr>
<td>1370</td>
</tr>
<tr>
<td>1380</td>
</tr>
<tr>
<td>1390</td>
</tr>
<tr>
<td>1400</td>
</tr>
<tr>
<td>1500</td>
</tr>
<tr>
<td>1510</td>
</tr>
<tr>
<td>1520</td>
</tr>
<tr>
<td>1530</td>
</tr>
<tr>
<td>1540</td>
</tr>
<tr>
<td>1550</td>
</tr>
<tr>
<td>1560</td>
</tr>
<tr>
<td>1570</td>
</tr>
<tr>
<td>1580</td>
</tr>
<tr>
<td>1590</td>
</tr>
<tr>
<td>1600</td>
</tr>
<tr>
<td>1610</td>
</tr>
<tr>
<td>1700</td>
</tr>
<tr>
<td>1710</td>
</tr>
<tr>
<td>1720</td>
</tr>
<tr>
<td>1730</td>
</tr>
<tr>
<td>1740</td>
</tr>
<tr>
<td>1750</td>
</tr>
<tr>
<td>1760</td>
</tr>
<tr>
<td>1770</td>
</tr>
<tr>
<td>1780</td>
</tr>
<tr>
<td>1790</td>
</tr>
<tr>
<td>1800</td>
</tr>
<tr>
<td>1810</td>
</tr>
<tr>
<td>1820</td>
</tr>
<tr>
<td>1900</td>
</tr>
<tr>
<td>1910</td>
</tr>
<tr>
<td>1920</td>
</tr>
<tr>
<td>1930</td>
</tr>
<tr>
<td>1940</td>
</tr>
<tr>
<td>1950</td>
</tr>
<tr>
<td>1960</td>
</tr>
<tr>
<td>Year</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>1970</td>
</tr>
<tr>
<td>1980</td>
</tr>
<tr>
<td>1990</td>
</tr>
<tr>
<td>2000</td>
</tr>
<tr>
<td>2010</td>
</tr>
<tr>
<td>2020</td>
</tr>
<tr>
<td>2030</td>
</tr>
<tr>
<td>2040</td>
</tr>
<tr>
<td>2050</td>
</tr>
<tr>
<td>2060</td>
</tr>
<tr>
<td>2070</td>
</tr>
<tr>
<td>2080</td>
</tr>
<tr>
<td>2090</td>
</tr>
<tr>
<td>2100</td>
</tr>
<tr>
<td>2110</td>
</tr>
<tr>
<td>2120</td>
</tr>
<tr>
<td>2130</td>
</tr>
<tr>
<td>2140</td>
</tr>
<tr>
<td>2150</td>
</tr>
<tr>
<td>2160</td>
</tr>
</tbody>
</table>
C.2 Tunnel module

<table>
<thead>
<tr>
<th>Code list of the _AbstractTunnel attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_class.xml">http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_class.xml</a></td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the _AbstractTunnel attributes function and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_function.xml">http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_function.xml</a></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_usage.xml">http://www.sig3d.org/codelists/standard/tunnel/2.0/_AbstractTunnel_usage.xml</a></td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1010</td>
</tr>
</tbody>
</table>
### C.3 Bridge module

<table>
<thead>
<tr>
<th>Code list of the _AbstractBridge attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_class.xml">http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_class.xml</a></td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1020</td>
</tr>
<tr>
<td>1030</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the _AbstractBridge attributes function and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_function.xml">http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_function.xml</a></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_usage.xml">http://www.sig3d.org/codelists/standard/bridge/2.0/_AbstractBridge_usage.xml</a></td>
</tr>
<tr>
<td>1000</td>
</tr>
<tr>
<td>1010</td>
</tr>
<tr>
<td>1030</td>
</tr>
</tbody>
</table>
C.4 CityFurniture module

### Code list of the CityFurniture attribute class

<table>
<thead>
<tr>
<th>Code</th>
<th>CityFurniture attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>traffic</td>
</tr>
<tr>
<td>1010</td>
<td>communication</td>
</tr>
</tbody>
</table>

### Code list of the CityFurniture attributes function and usage

<table>
<thead>
<tr>
<th>Code</th>
<th>CityFurniture attributes function and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>communication fixture</td>
</tr>
<tr>
<td>1010</td>
<td>telephone box</td>
</tr>
<tr>
<td>1020</td>
<td>postbox</td>
</tr>
<tr>
<td>1030</td>
<td>emergency call fixture</td>
</tr>
<tr>
<td>1040</td>
<td>fire detector</td>
</tr>
<tr>
<td>1050</td>
<td>police call post</td>
</tr>
<tr>
<td>1060</td>
<td>switching unit</td>
</tr>
<tr>
<td>1070</td>
<td>road sign</td>
</tr>
<tr>
<td>1080</td>
<td>traffic light</td>
</tr>
<tr>
<td>1090</td>
<td>free-standing sign</td>
</tr>
<tr>
<td>1100</td>
<td>free-standing warning sign</td>
</tr>
<tr>
<td>1110</td>
<td>bus stop</td>
</tr>
<tr>
<td>1120</td>
<td>milestone</td>
</tr>
<tr>
<td>1130</td>
<td>rail level crossing</td>
</tr>
<tr>
<td>1140</td>
<td>gate</td>
</tr>
<tr>
<td>1150</td>
<td>streetlamp, lantern or candelabra</td>
</tr>
<tr>
<td>1160</td>
<td>column</td>
</tr>
<tr>
<td>1170</td>
<td>lamp post</td>
</tr>
<tr>
<td>1180</td>
<td>flagpole</td>
</tr>
<tr>
<td>1190</td>
<td>street sink box</td>
</tr>
<tr>
<td>1200</td>
<td>rubbish bin</td>
</tr>
<tr>
<td>1210</td>
<td>clock</td>
</tr>
<tr>
<td>1220</td>
<td>directional spot light</td>
</tr>
<tr>
<td>1230</td>
<td>floodlight mast</td>
</tr>
<tr>
<td>1240</td>
<td>windmill</td>
</tr>
<tr>
<td>1250</td>
<td>solar cell</td>
</tr>
<tr>
<td>1260</td>
<td>water wheel</td>
</tr>
</tbody>
</table>

http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_class.xml
http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_function.xml
http://www.sig3d.org/codelists/standard/cityfurniture/2.0/CityFurniture_usage.xml
C.5 LandUse module

<table>
<thead>
<tr>
<th>Code list of the LandUse attribute class</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_class.xml">http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_class.xml</a></td>
</tr>
<tr>
<td>1000 Settlement Area</td>
</tr>
<tr>
<td>1100 Undeveloped Area</td>
</tr>
<tr>
<td>2000 Traffic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code list of the LandUse attributes function and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_function.xml">http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_function.xml</a></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_usage.xml">http://www.sig3d.org/codelists/standard/landuse/2.0/LandUse_usage.xml</a></td>
</tr>
<tr>
<td>1010 Residential</td>
</tr>
<tr>
<td>1020 Industry and Business</td>
</tr>
<tr>
<td>1030 Mixed use</td>
</tr>
<tr>
<td>1040 Special Function Area</td>
</tr>
<tr>
<td>1050 Monument</td>
</tr>
<tr>
<td>1060 Dump</td>
</tr>
<tr>
<td>1070 Mining</td>
</tr>
<tr>
<td>1110 Park</td>
</tr>
<tr>
<td>1120 Cemetery</td>
</tr>
<tr>
<td>1130 Sports, leisure and recreation</td>
</tr>
<tr>
<td>1140 Open pit, quarry</td>
</tr>
<tr>
<td>2010 Road</td>
</tr>
<tr>
<td>2020 Railway</td>
</tr>
<tr>
<td>2030 Airfield</td>
</tr>
<tr>
<td>2040 Shipping</td>
</tr>
</tbody>
</table>
C.6 Mime types

<table>
<thead>
<tr>
<th>Codelist of the ImplicitGeometry (Core module) / _Texture (Appearance module) attribute mimeType</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/core/2.0/ImplicitGeometry_mimeType.xml">http://www.sig3d.org/codelists/standard/core/2.0/ImplicitGeometry_mimeType.xml</a></td>
</tr>
<tr>
<td><a href="http://www.sig3d.org/codelists/standard/appearance/2.0/_Texture_mimeType.xml">http://www.sig3d.org/codelists/standard/appearance/2.0/_Texture_mimeType.xml</a></td>
</tr>
</tbody>
</table>

The MIME types given in this 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. This code list is not exhaustive. It contains a selection of frequently used MIME types.

- model/vrml: VRML97
- application/x-3ds: 3ds max
- application/dxf: AutoCad DXF
- application/x-autocad: AutoCad DXF
- application/x-dxf: AutoCad DXF
- application/acad: AutoCad DWG
- application/x-shockwave-flash: Shockwave 3D
- model/x3d+vrml: X3D

The MIME types are listed as follows:

- model/x3d+xml: X3D
- model/x3d+binary: X3D
- image/gif: *.gif images
- image/jpeg: *.jpeg, *.jpg images
- image/png: *.png images
- application/x-dwg: AutoCad DWG
- image/tiff: *.tiff, *.tif images
- image/bmp: *.bmp images
## C.7 Vegetation module

### Code list of the SolitaryVegetationObject attribute class

<table>
<thead>
<tr>
<th>Code</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>shrub</td>
<td>1060</td>
</tr>
<tr>
<td>1010</td>
<td>low plants</td>
<td>1070</td>
</tr>
<tr>
<td>1020</td>
<td>medium high plants</td>
<td>1080</td>
</tr>
<tr>
<td>1030</td>
<td>high plants</td>
<td>1090</td>
</tr>
<tr>
<td>1040</td>
<td>grasses</td>
<td>1100</td>
</tr>
<tr>
<td>1050</td>
<td>ferns</td>
<td>9999</td>
</tr>
</tbody>
</table>

### Code list of the SolitaryVegetationObject attributes function and usage

http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_function.xml
http://www.sig3d.org/codelists/standard/vegetation/2.0/SolitaryVegetationObject_usage.xml

Code lists are identically with SolitaryVegetationObject class

### Code list of the PlantCover attribute class

<table>
<thead>
<tr>
<th>Code</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>Lemnetea</td>
<td>1280</td>
</tr>
<tr>
<td>1020</td>
<td>Asplenietea rupestris</td>
<td>1290</td>
</tr>
<tr>
<td>1030</td>
<td>Adiantetea</td>
<td>1300</td>
</tr>
<tr>
<td>1040</td>
<td>Thlaspietea rotundifolii</td>
<td>1310</td>
</tr>
<tr>
<td>1050</td>
<td>Crithmo-Limonietea</td>
<td>1320</td>
</tr>
<tr>
<td>1060</td>
<td>Ammophieta</td>
<td>1330</td>
</tr>
<tr>
<td>1070</td>
<td>Cakiletea maritimae halophile</td>
<td>1340</td>
</tr>
<tr>
<td>1080</td>
<td>Secalinetea</td>
<td>1350</td>
</tr>
<tr>
<td>1090</td>
<td>Chenopodietea</td>
<td>1360</td>
</tr>
<tr>
<td>1100</td>
<td>Onopordetea</td>
<td>1370</td>
</tr>
<tr>
<td>1110</td>
<td>Epilobietea angustifolii</td>
<td>1380</td>
</tr>
<tr>
<td>1120</td>
<td>Bidentetea tripartiti</td>
<td>1390</td>
</tr>
<tr>
<td>1130</td>
<td>Zoosteretea marinae halophile</td>
<td>1400</td>
</tr>
<tr>
<td>1140</td>
<td>Ruppietea maritimae</td>
<td>1410</td>
</tr>
<tr>
<td>1150</td>
<td>Potametea haftende</td>
<td>1420</td>
</tr>
<tr>
<td>1160</td>
<td>Litorelletea</td>
<td>1430</td>
</tr>
<tr>
<td>1170</td>
<td>Plantaginetea majoris</td>
<td>1440</td>
</tr>
<tr>
<td>1180</td>
<td>Isoetinanojuncetea</td>
<td>1450</td>
</tr>
<tr>
<td>1190</td>
<td>Montino-Cardaminetea</td>
<td>1460</td>
</tr>
<tr>
<td>1200</td>
<td>Corynephoretea</td>
<td>1470</td>
</tr>
<tr>
<td>1210</td>
<td>Asteretea tripolium</td>
<td>1480</td>
</tr>
<tr>
<td>1220</td>
<td>Salicornietea</td>
<td>1490</td>
</tr>
<tr>
<td>1230</td>
<td>Juncetea maritimae</td>
<td>1500</td>
</tr>
<tr>
<td>1240</td>
<td>Phragmitetea</td>
<td>1510</td>
</tr>
<tr>
<td>1250</td>
<td>Spartinetea</td>
<td>1520</td>
</tr>
<tr>
<td>1260</td>
<td>Sedo-Scleranthetea</td>
<td>9999</td>
</tr>
<tr>
<td>1270</td>
<td>Saliceta herbaceae</td>
<td></td>
</tr>
</tbody>
</table>

### Code list of the PlantCover attributes function and usage

http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_function.xml
http://www.sig3d.org/codelists/standard/vegetation/2.0/PlantCover_usage.xml

Code lists are identically with PlantCover class
<table>
<thead>
<tr>
<th>Code</th>
<th>Species Name</th>
<th>Code</th>
<th>Species Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1640</td>
<td>Abies alba</td>
<td>1790</td>
<td>Acer circinatum</td>
</tr>
<tr>
<td>1650</td>
<td>Abies cephalonica</td>
<td>1800</td>
<td>Acer Davidii</td>
</tr>
<tr>
<td>1660</td>
<td>Abies concolor</td>
<td>1810</td>
<td>Acer ginnala Maxim</td>
</tr>
<tr>
<td>1670</td>
<td>Abies grandis</td>
<td>1820</td>
<td>Acer grosseri</td>
</tr>
<tr>
<td>1680</td>
<td>Abies homolepis</td>
<td>1830</td>
<td>Acer monspessulanum</td>
</tr>
<tr>
<td>1690</td>
<td>Abies koreana</td>
<td>1840</td>
<td>Acer negundo</td>
</tr>
<tr>
<td>1700</td>
<td>Abies lasiocarpa</td>
<td>1850</td>
<td>Acer palmatum</td>
</tr>
<tr>
<td>1710</td>
<td>Abies nordmanniana</td>
<td>1860</td>
<td>Acer platanoides</td>
</tr>
<tr>
<td>1720</td>
<td>Abies pinsapo</td>
<td>1870</td>
<td>Acer platanoides 'Crimson King'</td>
</tr>
<tr>
<td>1730</td>
<td>Abies procera</td>
<td>1880</td>
<td>Acer pseudoplatanus</td>
</tr>
<tr>
<td>1740</td>
<td>Abies procera 'Glaucia'</td>
<td>1890</td>
<td>Acer rubrum</td>
</tr>
<tr>
<td>1750</td>
<td>Abies veitchii</td>
<td>1900</td>
<td>Acer saccharinum</td>
</tr>
<tr>
<td>1760</td>
<td>Acer campéstre</td>
<td>1910</td>
<td>Acer saccharum Marsch</td>
</tr>
<tr>
<td>1770</td>
<td>Acer capillipes</td>
<td>1920</td>
<td>Acer tartaricum</td>
</tr>
<tr>
<td>1780</td>
<td>Acer cappadocicum</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
### C.8 Transportation module

#### Code list of the AuxiliaryTrafficArea attribute function

[http://www.sig3d.org/codelists/standard/transportation/2.0/AuxiliaryTrafficArea_function.xml](http://www.sig3d.org/codelists/standard/transportation/2.0/AuxiliaryTrafficArea_function.xml)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>soft shoulder</td>
</tr>
<tr>
<td>1010</td>
<td>hard shoulder</td>
</tr>
<tr>
<td>1020</td>
<td>green area</td>
</tr>
<tr>
<td>1030</td>
<td>middle lane</td>
</tr>
<tr>
<td>1040</td>
<td>lay by</td>
</tr>
<tr>
<td>1100</td>
<td>parking bay</td>
</tr>
<tr>
<td>1200</td>
<td>ditch</td>
</tr>
<tr>
<td>1210</td>
<td>drainage</td>
</tr>
<tr>
<td>1220</td>
<td>kerbstone</td>
</tr>
<tr>
<td>1230</td>
<td>flower tub</td>
</tr>
<tr>
<td>1300</td>
<td>traffic island</td>
</tr>
<tr>
<td>1400</td>
<td>bank</td>
</tr>
<tr>
<td>1410</td>
<td>embankment, dike</td>
</tr>
<tr>
<td>1420</td>
<td>railroad embankment</td>
</tr>
<tr>
<td>1430</td>
<td>noise protection</td>
</tr>
<tr>
<td>1440</td>
<td>noise protection wall</td>
</tr>
<tr>
<td>1500</td>
<td>noise guard bar</td>
</tr>
<tr>
<td>1600</td>
<td>towpath</td>
</tr>
<tr>
<td>1700</td>
<td>others</td>
</tr>
</tbody>
</table>

#### Code list of the TrafficArea attribute function

[http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_function.xml](http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_function.xml)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>driving_lane</td>
</tr>
<tr>
<td>2</td>
<td>footpath</td>
</tr>
<tr>
<td>3</td>
<td>cyclepath</td>
</tr>
<tr>
<td>4</td>
<td>combined foot-/cyclepath</td>
</tr>
<tr>
<td>5</td>
<td>square</td>
</tr>
<tr>
<td>6</td>
<td>car_park</td>
</tr>
<tr>
<td>7</td>
<td>parking_lay_by</td>
</tr>
<tr>
<td>8</td>
<td>rail</td>
</tr>
<tr>
<td>9</td>
<td>rail_road_combined</td>
</tr>
<tr>
<td>10</td>
<td>drainage</td>
</tr>
<tr>
<td>11</td>
<td>road marking</td>
</tr>
<tr>
<td>12</td>
<td>road_marking_direction</td>
</tr>
<tr>
<td>13</td>
<td>road_marking_lane</td>
</tr>
<tr>
<td>14</td>
<td>road_marking_restricted</td>
</tr>
<tr>
<td>15</td>
<td>road_marking_crosswalk</td>
</tr>
<tr>
<td>16</td>
<td>road_marking_stop</td>
</tr>
<tr>
<td>17</td>
<td>road_marking_other</td>
</tr>
<tr>
<td>18</td>
<td>overhead wire (trolley)</td>
</tr>
<tr>
<td>19</td>
<td>train platform</td>
</tr>
<tr>
<td>20</td>
<td>crosswalk</td>
</tr>
<tr>
<td>21</td>
<td>barrier</td>
</tr>
<tr>
<td>22</td>
<td>stairs</td>
</tr>
<tr>
<td>23</td>
<td>escalator</td>
</tr>
<tr>
<td>24</td>
<td>filtering lane</td>
</tr>
<tr>
<td>25</td>
<td>airport_runway</td>
</tr>
<tr>
<td>26</td>
<td>airport_taxiway</td>
</tr>
<tr>
<td>27</td>
<td>airport_apron</td>
</tr>
<tr>
<td>28</td>
<td>airport_heliport</td>
</tr>
<tr>
<td>29</td>
<td>airport_runway_marking</td>
</tr>
<tr>
<td>30</td>
<td>green spaces</td>
</tr>
<tr>
<td>31</td>
<td>recreation</td>
</tr>
<tr>
<td>32</td>
<td>bus_lay_by</td>
</tr>
<tr>
<td>33</td>
<td>motorway</td>
</tr>
<tr>
<td>34</td>
<td>motorway_entry</td>
</tr>
<tr>
<td>35</td>
<td>motorway_exit</td>
</tr>
<tr>
<td>36</td>
<td>motorway_emergency lane</td>
</tr>
<tr>
<td>37</td>
<td>private_area</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

#### Code list of the TrafficArea attribute usage

[http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_usage.xml](http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_usage.xml)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pedestrian</td>
</tr>
<tr>
<td>2</td>
<td>car</td>
</tr>
<tr>
<td>3</td>
<td>truck</td>
</tr>
<tr>
<td>4</td>
<td>bus, taxi</td>
</tr>
<tr>
<td>5</td>
<td>train</td>
</tr>
<tr>
<td>6</td>
<td>bicycle</td>
</tr>
<tr>
<td>7</td>
<td>motorcycle</td>
</tr>
<tr>
<td>8</td>
<td>tram, streetcar</td>
</tr>
<tr>
<td>9</td>
<td>boat, ferry, ship</td>
</tr>
<tr>
<td>10</td>
<td>teleferic</td>
</tr>
<tr>
<td>11</td>
<td>aeroplane</td>
</tr>
<tr>
<td>12</td>
<td>helicopter</td>
</tr>
<tr>
<td>13</td>
<td>taxi</td>
</tr>
<tr>
<td>14</td>
<td>horse</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>
## Code list of the TrafficArea and AuxiliaryTrafficArea attribute surfaceMaterial

http://www.sig3d.org/codelists/standard/transportation/2.0/TrafficArea_surfaceMaterial.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Code</th>
<th>Name</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>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

## Code list of the TransportationComplex attribute class

http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_class.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Class</th>
<th>Code</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>private</td>
<td>1050</td>
<td>air traffic</td>
</tr>
<tr>
<td>1010</td>
<td>common</td>
<td>1060</td>
<td>rail traffic</td>
</tr>
<tr>
<td>1020</td>
<td>civil</td>
<td>1070</td>
<td>waterway</td>
</tr>
<tr>
<td>1030</td>
<td>military</td>
<td>1080</td>
<td>subway</td>
</tr>
<tr>
<td>1040</td>
<td>road traffic</td>
<td>1090</td>
<td>others</td>
</tr>
</tbody>
</table>

## Code list of the TransportationComplex attributes function and usage

http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_function.xml
http://www.sig3d.org/codelists/standard/transportation/2.0/TransportationComplex_usage.xml

<table>
<thead>
<tr>
<th>Code</th>
<th>Function</th>
<th>Code</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>road</td>
<td>1855</td>
<td>railway track</td>
</tr>
<tr>
<td>1010</td>
<td>freeway/motorway</td>
<td>1860</td>
<td>magnetic levitation train</td>
</tr>
<tr>
<td>1020</td>
<td>highway/national primary road</td>
<td>1900</td>
<td>railway station</td>
</tr>
<tr>
<td>1030</td>
<td>land road</td>
<td>1910</td>
<td>stop</td>
</tr>
<tr>
<td>1040</td>
<td>district road</td>
<td>1920</td>
<td>station</td>
</tr>
<tr>
<td>1050</td>
<td>municipal road</td>
<td>2000</td>
<td>power-wheel</td>
</tr>
<tr>
<td>1060</td>
<td>main through-road</td>
<td>2100</td>
<td>airport</td>
</tr>
<tr>
<td>1100</td>
<td>freeway interchange/ highway junction</td>
<td>2110</td>
<td>international airport</td>
</tr>
<tr>
<td>1110</td>
<td>junction</td>
<td>2120</td>
<td>regional airport</td>
</tr>
<tr>
<td>1200</td>
<td>road</td>
<td>2130</td>
<td>landing place</td>
</tr>
<tr>
<td>1210</td>
<td>driveway</td>
<td>2140</td>
<td>heliport</td>
</tr>
<tr>
<td>1220</td>
<td>footpath/footway</td>
<td>2150</td>
<td>landing place</td>
</tr>
<tr>
<td>1230</td>
<td>hiking trail</td>
<td>2160</td>
<td>gliding airfield</td>
</tr>
<tr>
<td>1240</td>
<td>bikeway/cycle-path</td>
<td>2170</td>
<td>taxiway</td>
</tr>
<tr>
<td>1250</td>
<td>bridleway/bridlepath</td>
<td>2180</td>
<td>apron</td>
</tr>
<tr>
<td>1260</td>
<td>main agricultural road</td>
<td>2190</td>
<td>runaway</td>
</tr>
<tr>
<td>1270</td>
<td>agricultural road</td>
<td>2200</td>
<td>canal</td>
</tr>
<tr>
<td>1280</td>
<td>bikeway/footway</td>
<td>2300</td>
<td>harbor</td>
</tr>
<tr>
<td>1290</td>
<td>access road</td>
<td>2310</td>
<td>pleasure craft harbour</td>
</tr>
<tr>
<td>1300</td>
<td>dead-end road</td>
<td>2400</td>
<td>ferry</td>
</tr>
<tr>
<td>1400</td>
<td>lane</td>
<td>2410</td>
<td>car ferry</td>
</tr>
<tr>
<td>1410</td>
<td>lane, one direction</td>
<td>2420</td>
<td>train ferry</td>
</tr>
<tr>
<td>1420</td>
<td>lane, both direction</td>
<td>2430</td>
<td>ferry</td>
</tr>
<tr>
<td>1500</td>
<td>pedestrian zone</td>
<td>2500</td>
<td>landing stage</td>
</tr>
<tr>
<td>1600</td>
<td>place</td>
<td>2600</td>
<td>waterway I order</td>
</tr>
<tr>
<td>1610</td>
<td>parking area</td>
<td>2610</td>
<td>navigable river</td>
</tr>
<tr>
<td>1620</td>
<td>marketplace</td>
<td>2620</td>
<td>inland navigation waterway 0</td>
</tr>
<tr>
<td>1700</td>
<td>service area</td>
<td>2621</td>
<td>inland navigation waterway 0</td>
</tr>
<tr>
<td>1800</td>
<td>rail transport</td>
<td>2622</td>
<td>inland navigation waterway I</td>
</tr>
<tr>
<td>1805</td>
<td>rail</td>
<td>2623</td>
<td>inland navigation waterway II</td>
</tr>
<tr>
<td>1810</td>
<td>urban/city train</td>
<td>2624</td>
<td>inland navigation waterway III</td>
</tr>
<tr>
<td>Year</td>
<td>Description</td>
<td>Code</td>
<td>Type</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------</td>
<td>-------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>1815</td>
<td>city railway</td>
<td>2625</td>
<td>inland navigation waterway IV</td>
</tr>
<tr>
<td>1820</td>
<td>tram</td>
<td>2626</td>
<td>inland navigation waterway V</td>
</tr>
<tr>
<td>1825</td>
<td>subway</td>
<td>2627</td>
<td>inland navigation waterway VI</td>
</tr>
<tr>
<td>1830</td>
<td>funicular/mountain railway</td>
<td>2628</td>
<td>inland navigation waterway VII</td>
</tr>
<tr>
<td>1835</td>
<td>mountain railway</td>
<td>2630</td>
<td>maritime navigation</td>
</tr>
<tr>
<td>1840</td>
<td>chairlift</td>
<td>2640</td>
<td>navigable lake</td>
</tr>
<tr>
<td>1845</td>
<td>ski-lift/ski tow lift</td>
<td>2700</td>
<td>others</td>
</tr>
<tr>
<td>1850</td>
<td>suspension railway</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## C.9 WaterBody module

### Code list of the WaterBody attribute class

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>sea</td>
</tr>
<tr>
<td>1010</td>
<td>tidal waterbody</td>
</tr>
<tr>
<td>1020</td>
<td>watercourse</td>
</tr>
<tr>
<td>1030</td>
<td>river / stream</td>
</tr>
<tr>
<td>1040</td>
<td>ditch</td>
</tr>
<tr>
<td>1050</td>
<td>spring / water hole</td>
</tr>
<tr>
<td>1060</td>
<td>lake / pond</td>
</tr>
<tr>
<td>1070</td>
<td>bayou</td>
</tr>
<tr>
<td>1080</td>
<td>body of standing water</td>
</tr>
<tr>
<td>1090</td>
<td>waterfall</td>
</tr>
<tr>
<td>1100</td>
<td>rapids</td>
</tr>
<tr>
<td>1110</td>
<td>swamp</td>
</tr>
<tr>
<td>1120</td>
<td>sinkhole (karst)</td>
</tr>
<tr>
<td>1130</td>
<td>ephemeral watercourse</td>
</tr>
<tr>
<td>1014</td>
<td>flooded land</td>
</tr>
<tr>
<td>1150</td>
<td>artificial waterbody</td>
</tr>
<tr>
<td>1160</td>
<td>aqueduct</td>
</tr>
<tr>
<td>1170</td>
<td>canal</td>
</tr>
<tr>
<td>1180</td>
<td>port basin</td>
</tr>
<tr>
<td>1190</td>
<td>reservoir</td>
</tr>
<tr>
<td>1200</td>
<td>excavation pond</td>
</tr>
<tr>
<td>1210</td>
<td>moat</td>
</tr>
<tr>
<td>1220</td>
<td>pool</td>
</tr>
<tr>
<td>1230</td>
<td>fountain</td>
</tr>
<tr>
<td>1240</td>
<td>well</td>
</tr>
<tr>
<td>1250</td>
<td>cistern</td>
</tr>
<tr>
<td>1260</td>
<td>fish ladder</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### Code list of the WaterBody attribute function

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>nature-sanctuary</td>
</tr>
<tr>
<td>1010</td>
<td>protected waterbody</td>
</tr>
<tr>
<td>1020</td>
<td>reservoir</td>
</tr>
<tr>
<td>1030</td>
<td>retention waterbody</td>
</tr>
<tr>
<td>1040</td>
<td>flood plain waterbody</td>
</tr>
<tr>
<td>1050</td>
<td>waterway</td>
</tr>
<tr>
<td>1060</td>
<td>harbor waterbody</td>
</tr>
<tr>
<td>1070</td>
<td>sluice waterbody</td>
</tr>
<tr>
<td>1080</td>
<td>sewage system</td>
</tr>
<tr>
<td>1090</td>
<td>public swimming</td>
</tr>
<tr>
<td>1100</td>
<td>public fountain</td>
</tr>
<tr>
<td>1110</td>
<td>private waterbody</td>
</tr>
<tr>
<td>1120</td>
<td>irrigation waterbody</td>
</tr>
<tr>
<td>1130</td>
<td>watering place</td>
</tr>
<tr>
<td>1140</td>
<td>industrial waterbody</td>
</tr>
<tr>
<td>1150</td>
<td>waterbody for fire-fighting</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### Code list of the WaterBody attribute usage

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>sanctuary</td>
</tr>
<tr>
<td>1010</td>
<td>recreation / sports</td>
</tr>
<tr>
<td>1020</td>
<td>drinking water supply</td>
</tr>
<tr>
<td>1030</td>
<td>hydroelectric water supply</td>
</tr>
<tr>
<td>1040</td>
<td>ocean shipping</td>
</tr>
<tr>
<td>1050</td>
<td>inland shipping</td>
</tr>
<tr>
<td>1060</td>
<td>sewer</td>
</tr>
<tr>
<td>1070</td>
<td>port</td>
</tr>
<tr>
<td>1080</td>
<td>anchorage</td>
</tr>
<tr>
<td>1090</td>
<td>public use</td>
</tr>
<tr>
<td>1100</td>
<td>private use</td>
</tr>
<tr>
<td>1110</td>
<td>industrial / craft water supply</td>
</tr>
<tr>
<td>1120</td>
<td>military use</td>
</tr>
<tr>
<td>1130</td>
<td>mining / excavation</td>
</tr>
<tr>
<td>1140</td>
<td>irrigation water supply</td>
</tr>
<tr>
<td>1150</td>
<td>fishing water</td>
</tr>
<tr>
<td>1160</td>
<td>fish farm</td>
</tr>
<tr>
<td>1170</td>
<td>archaeological site</td>
</tr>
<tr>
<td>1180</td>
<td>water protection area</td>
</tr>
<tr>
<td>1190</td>
<td>abandoned</td>
</tr>
<tr>
<td>9999</td>
<td>unknown</td>
</tr>
</tbody>
</table>

### Code list of the WaterSurface attribute waterLevel

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>MSL - Mean Sea Level</td>
</tr>
<tr>
<td>1010</td>
<td>LAT - Lowest Astronomical Tide</td>
</tr>
<tr>
<td>1020</td>
<td>National Water Level</td>
</tr>
<tr>
<td>1090</td>
<td>Hundred Year Flood</td>
</tr>
<tr>
<td>1100</td>
<td>highest known water level</td>
</tr>
<tr>
<td>1110</td>
<td>critical low-water level</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>1030</td>
<td>Mean High Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1040</td>
<td>Extreme High Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1050</td>
<td>Mean Low Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1060</td>
<td>Extreme Low Tide (related to National Waterlevel)</td>
</tr>
<tr>
<td>1070</td>
<td>Mean Water Level (watercourse)</td>
</tr>
<tr>
<td>1080</td>
<td>critical high-water level</td>
</tr>
</tbody>
</table>
## C.10 CityObjectGroup module

### Code list of the CityObjectGroup attribute class

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>building separation</td>
</tr>
<tr>
<td>2000</td>
<td>assembly</td>
</tr>
</tbody>
</table>

### Code list of the CityObjectGroup attributes function and usage

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>lod1Storey</td>
</tr>
<tr>
<td>1010</td>
<td>lod2Storey</td>
</tr>
<tr>
<td>1020</td>
<td>lod3Storey</td>
</tr>
<tr>
<td>1030</td>
<td>lod4Storey</td>
</tr>
</tbody>
</table>

[http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_class.xml](http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_class.xml)

[http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_function.xml](http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_function.xml)

[http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_usage.xml](http://www.sig3d.org/codelists/standard/cityobjectgroup/2.0/CityObjectGroup_usage.xml)
Annex D
(informative)

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>_Surface</td>
<td>CompositeSolid (boundary is restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td>_Surface</td>
<td>Polygon (with holes, modelled by _Rings. The boundary is restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td>_Surface</td>
<td>OrientableSurface (defined in CityGML’s TexturedSurface module, not in GML. This modelling approach has been marked deprecated)</td>
</tr>
<tr>
<td>_Surface</td>
<td>CompositeSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td>_Surface</td>
<td>TriangulatedSurface</td>
</tr>
<tr>
<td>_Surface</td>
<td>TIN</td>
</tr>
<tr>
<td>_Curve</td>
<td>LineString</td>
</tr>
<tr>
<td>_Curve</td>
<td>CompositeCurve (members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td>_Coverage</td>
<td>RectifiedGridCoverage</td>
</tr>
<tr>
<td>_AbstractGeometricAggregate</td>
<td>MultiSolid</td>
</tr>
<tr>
<td>_AbstractGeometricAggregate</td>
<td>MultiSurface (members are restricted to OrientableSurfaces, TexturedSurfaces, Polygons or CompositeSurfaces)</td>
</tr>
<tr>
<td>_AbstractGeometricAggregate</td>
<td>MultiCurve (members are restricted to LineStrings or CompositeCurves)</td>
</tr>
<tr>
<td>_AbstractGeometricAggregate</td>
<td>MultiPoint</td>
</tr>
<tr>
<td>_AbstractGeometricAggregate</td>
<td>GeometricComplex (restricted to connected, linear networks)</td>
</tr>
</tbody>
</table>
Annex E
(informative)

Overview of the assignment of features to LODs

The following table lists all feature types of CityGML. For each feature type, all non-spatial and spatial properties are given including their type. Each feature is assigned a range of LOD in which it may occur, and for each property the LOD in which it represents the feature is stated. Each feature type’s name is preceded by a prefix indicating the CityGML module defining the feature type. For a list of prefixes and associated CityGML modules, please refer to chapter 4.3 and chapter 7.

<table>
<thead>
<tr>
<th>Feature Class</th>
<th>Property</th>
<th>Type</th>
<th>LOD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CityGML Core module</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>core.CityModelType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cityObjectMember</td>
<td>gml:FeaturePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:appearanceMember</td>
<td>app:AppearancePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfCityModel</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AbstractCityObjectType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>creationDate</td>
<td>xs:date</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>terminationDate</td>
<td>xs:date</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>externalReference</td>
<td>core:ExternalReferenceType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>generalizesTo</td>
<td>core:GeneralizationRelationType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>relativeToTerrain</td>
<td>core:RelativeToTerrainType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>relativeToWater</td>
<td>core:RelativeToWaterType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:appearance</td>
<td>app:AppearancePropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>gen:_genericAttribute</td>
<td>gen:AbstractGenericAttributeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfCityObject</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AbstractSiteType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfSite</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>core:AddressType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xalAddress</td>
<td>core:xalAddressPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>multiPoint</td>
<td>gml:MultiPointPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfAddress</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td><strong>Appearance module</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app:AppearanceType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>theme</td>
<td>xs:string</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>surfaceDataMember</td>
<td>app:SurfaceDataPropertyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfAppearance</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:AbstractSurfaceDataType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isFront</td>
<td>xs:Boolean</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfSurfaceData</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:AbstractTextureType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imageURI</td>
<td>xs:anyURI</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>mimeType</td>
<td>gml:CodeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>textureType</td>
<td>app:TextureTypeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>wrapMode</td>
<td>app:WrapModeType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>borderColor</td>
<td>app:ColorPlusOpacity</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfTexture</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>app:ParameterizedTextureType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>app:TextureAssociationType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td>_Generic:ApplicationPropertyOfParameterizedTexture</td>
<td>xs:anyType</td>
<td>0 – 4</td>
<td></td>
</tr>
<tr>
<td><strong>app:GeoreferencedTextureType</strong></td>
<td></td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>preferWorldFile</td>
<td>xs:boolean</td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>referencePoint</td>
<td>gml:PointPropertyType</td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>orientation</td>
<td>core:TransformationMatrix2x2Type</td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>target</td>
<td>xs:anyURI</td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
<td></td>
</tr>
<tr>
<td>OfGeoreferencedTexture</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>app:X3DMaterialType</strong></th>
<th></th>
<th>0 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambientIntensity</td>
<td>core:doubleBetween0and1</td>
<td>0 - 4</td>
</tr>
<tr>
<td>diffuseColor</td>
<td>app:Color</td>
<td>0 - 4</td>
</tr>
<tr>
<td>emissiveColor</td>
<td>app:Color</td>
<td>0 - 4</td>
</tr>
<tr>
<td>specularColor</td>
<td>app:Color</td>
<td>0 - 4</td>
</tr>
<tr>
<td>shininess</td>
<td>core:doubleBetween0and1</td>
<td>0 - 4</td>
</tr>
<tr>
<td>transparency</td>
<td>core:doubleBetween0and1</td>
<td>0 - 4</td>
</tr>
<tr>
<td>isSmooth</td>
<td>xs:boolean</td>
<td>0 - 4</td>
</tr>
<tr>
<td>target</td>
<td>xs:anyURI</td>
<td>0 - 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>OfX3DMaterial</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Building module</strong></th>
<th></th>
<th>0 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>OfAbstractBuilding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bldg:AbstractBuildingType</strong></th>
<th></th>
<th>0 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>yearOfConstruction</td>
<td>xs:gYear</td>
<td>0 - 4</td>
</tr>
<tr>
<td>yearOfDemolition</td>
<td>xs:gYear</td>
<td>0 - 4</td>
</tr>
<tr>
<td>roofType</td>
<td>gml:CodeType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>measuredHeight</td>
<td>gml:LengthType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>storeysAboveGround</td>
<td>xs:nonNegativeInteger</td>
<td>0 - 4</td>
</tr>
<tr>
<td>storeysBelowGround</td>
<td>xs:nonNegativeInteger</td>
<td>0 - 4</td>
</tr>
<tr>
<td>storeyHeightsAboveGround</td>
<td>gml:MeasureOrNullListType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>storeyHeightsBelowGround</td>
<td>gml:MeasureOrNullListType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>lod0FootPrint</td>
<td>gml:MultiSurfacePropertyType</td>
<td>0</td>
</tr>
<tr>
<td>lod0RoofEdge</td>
<td>gml:MultiSurfacePropertyType</td>
<td>0</td>
</tr>
<tr>
<td>lod1Solid</td>
<td>gml:SolidPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2Solid</td>
<td>gml:SolidPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>outerBuildingInstallation</td>
<td>bldg:BuildingInstallationPropertyType</td>
<td>2 - 4</td>
</tr>
<tr>
<td>interiorBuildingInstallation</td>
<td>bldg:IntBuildingInstallationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType</td>
<td>2 - 4</td>
</tr>
<tr>
<td>lod3Solid</td>
<td>gml:SolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>interiorRoom</td>
<td>bldg:InteriorRoomPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>consistsOfBuildingPart</td>
<td>bldg:BuildingPartPropertyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>address</td>
<td>core:AddressPropertyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>OfAbstractBuilding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bldg:BuildingType</strong></th>
<th></th>
<th>0 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>OfBuilding</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bldg:BuildingPartType</strong></th>
<th></th>
<th>0 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>_GenericApplicationProperty</td>
<td>xs:anyType</td>
<td>0 - 4</td>
</tr>
<tr>
<td>OfBuildingPart</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>bldg:BuildingInstallationType</strong></th>
<th></th>
<th>2 - 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>2 - 4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>2 - 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>2 – 4</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>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBuildingInstallation</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:IntBuildingInstallationType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td></td>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBuildingInstallation</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:AbstractBoundarySurfaceType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>opening</td>
<td>bldg:OpeningPropertyType</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBoundarySurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:RoofSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfRoofSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:WallSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfWallSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:OuterCeilingSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfOuterCeilingSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:OuterFloorSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfOuterFloorSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:GroundSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfGroundSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:ClosureSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfClosureSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:FloorSurfaceType</td>
<td></td>
<td>2 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfFloorSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:InteriorWallSurfaceType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfInteriorWallSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:CeilingSurfaceType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfCeilingSurface</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:AbstractOpeningType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td></td>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfOpening</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:WindowType</td>
<td></td>
<td>3 – 4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfWindow</td>
<td>xs:anyType</td>
</tr>
<tr>
<td>bldg:DoorType</td>
<td></td>
<td>3 – 4</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Usage</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>_GenericApplicationProperty Of Door</td>
<td>xs:anyType</td>
<td>3–4</td>
</tr>
<tr>
<td>bldg:RoomType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>interiorFurniture</td>
<td>bldg:InteriorFurniturePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>roomInstallation</td>
<td>bldg:IntBuildingInstallationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationProperty Of Room</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>bldg:BuildingFurnitureType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfBuildingFurniture</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>Bridge module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:AbstractBridgeType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>yearOfConstruction</td>
<td>xs:gYear</td>
<td>4</td>
</tr>
<tr>
<td>yearOfDemolition</td>
<td>xs:gYear</td>
<td>4</td>
</tr>
<tr>
<td>isMovable</td>
<td>xs:boolean</td>
<td>4</td>
</tr>
<tr>
<td>lod1Solid</td>
<td>gml:SolidPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2Solid</td>
<td>gml:SolidPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>outerBridgeConstruction</td>
<td>bldg:BridgeConstructionElementPropertyType</td>
<td>1–4</td>
</tr>
<tr>
<td>outerBridgeInstallation</td>
<td>bldg:BridgeInstallationPropertyType</td>
<td>2–4</td>
</tr>
<tr>
<td>interiorBridgeInstallation</td>
<td>bldg:IntBridgeInstallationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>bldg:BoundarySurfacePropertyType</td>
<td>2–4</td>
</tr>
<tr>
<td>lod3Solid</td>
<td>gml:SolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>interiorBridgeRoom</td>
<td>bldg:InteriorBridgeRoomPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>consistsOfBridgePart</td>
<td>bldg:BridgePartPropertyType</td>
<td>1–4</td>
</tr>
<tr>
<td>address</td>
<td>core:AddressPropertyType</td>
<td>1–4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfAbstractBridge</td>
<td>xs:anyType</td>
<td>1–4</td>
</tr>
<tr>
<td>bldg:BridgeType</td>
<td></td>
<td>1–4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfBridge</td>
<td>xs:anyType</td>
<td>1–4</td>
</tr>
<tr>
<td>bldg:BridgePartType</td>
<td></td>
<td>1–4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfBridgePart</td>
<td>xs:anyType</td>
<td>1–4</td>
</tr>
<tr>
<td>bldg:BridgeConstructionElementType</td>
<td></td>
<td>2–4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------</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>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>brid:BoundarySurfacePropertyType</td>
<td>2 – 4</td>
</tr>
</tbody>
</table>

_GenericApplicationProperty OfBridgeConstructionElement xs:anyType 2 – 4

brid:BridgeInstallationType

<table>
<thead>
<tr>
<th>class</th>
<th>gml:CodeType</th>
<th>2 – 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>2 – 4</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>core:ImplicitRepresentationPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>brid:BoundarySurfacePropertyType</td>
<td>2 – 4</td>
</tr>
</tbody>
</table>

_GenericApplicationProperty OfBridgeInstallation xs:anyType 2 – 4

brid:IntBridgeInstallationType

<table>
<thead>
<tr>
<th>class</th>
<th>gml:CodeType</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>brid:BoundarySurfacePropertyType</td>
<td>4</td>
</tr>
</tbody>
</table>

_GenericApplicationProperty OfIntBridgeInstallation xs:anyType 4

brid:AbstractBoundarySurfaceType

| lod2MultiSurface | gml:MultiSurfacePropertyType | 2 |
| lod3MultiSurface | gml:MultiSurfacePropertyType | 3 |
| lod4MultiSurface | gml:MultiSurfacePropertyType | 4 |
| opening | brid:OpeningPropertyType | 3 – 4 |

_GenericApplicationProperty OfBoundarySurface xs:anyType 2 – 4

brid:RoofSurfaceType

_GenericApplicationProperty OfRoofSurface xs:anyType 2 – 4

brid:WallSurfaceType

_GenericApplicationProperty OfWallSurface xs:anyType 2 – 4

brid:OuterCeilingSurfaceType

_GenericApplicationProperty OfOuterCeilingSurface xs:anyType 2 – 4

brid:OuterFloorSurfaceType

_GenericApplicationProperty OfOuterFloorSurface xs:anyType 2 – 4

brid:GroundSurfaceType

_GenericApplicationProperty OfGroundSurface xs:anyType 2 – 4

brid:ClosureSurfaceType

_GenericApplicationProperty OfClosureSurface xs:anyType 2 – 4
<table>
<thead>
<tr>
<th>Bridging Type</th>
<th>Property Description</th>
<th>XML Type</th>
<th>LODs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FloorSurfaceType</strong></td>
<td>_GenericApplicationProperty OfFloorSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>InteriorWallSurfaceType</strong></td>
<td>_GenericApplicationProperty OfInteriorWallSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>CeilingSurfaceType</strong></td>
<td>_GenericApplicationProperty OfCeilingSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>AbstractOpeningType</strong></td>
<td>lod3MultiSurface gml:MultiSurfacePropertyType</td>
<td>3</td>
<td>3–4</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface gml:MultiSurfacePropertyType</td>
<td>4</td>
<td>3–4</td>
</tr>
<tr>
<td></td>
<td>lod3ImplicitRepresentation core:ImplicitRepresentationPropertyType</td>
<td>3</td>
<td>3–4</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation core:ImplicitRepresentationPropertyType</td>
<td>4</td>
<td>3–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfOpening</td>
<td>xs:anyType</td>
<td>3–4</td>
</tr>
<tr>
<td><strong>WindowType</strong></td>
<td>_GenericApplicationProperty OfWindow</td>
<td>xs:anyType</td>
<td>3–4</td>
</tr>
<tr>
<td><strong>DoorType</strong></td>
<td>address core:AddressPropertyType</td>
<td>3–4</td>
<td>3–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfDoor</td>
<td>xs:anyType</td>
<td>3–4</td>
</tr>
<tr>
<td><strong>BridgeRoomType</strong></td>
<td>class gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>function gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>usage gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod4Solid gml:SolidPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface gml:MultiSurfacePropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>boundedBy brid:BoundarySurfacePropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>interiorFurniture brid:InteriorFurniturePropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>bridgeRoomInstallation brid:IntBridgeInstallationPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfBridgeRoom</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>BridgeFurnitureType</strong></td>
<td>class gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>function gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>usage gml:CodeType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod4Geometry gml:GeometryPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation core:ImplicitRepresentationPropertyType</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfBridgeFurniture</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>Relief module</strong></td>
<td>dem:ReliefFeatureType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>lod core:integerBetween0and4</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>reliefComponent dem:ReliefComponentPropertyType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfReliefFeature</td>
<td>xs:anyType</td>
<td>0–4</td>
</tr>
<tr>
<td><strong>AbstractReliefComponentType</strong></td>
<td>dem:AbstractReliefComponentType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>lod core:integerBetween0and4</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>extent gml:PolygonPropertyType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfReliefComponent</td>
<td>xs:anyType</td>
<td>0–4</td>
</tr>
<tr>
<td><strong>TINReliefType</strong></td>
<td>dem:TINReliefType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>tin dem:tinPropertyType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfTINRelief</td>
<td>xs:anyType</td>
<td>0–4</td>
</tr>
<tr>
<td><strong>RasterReliefType</strong></td>
<td>dem:RasterReliefType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>grid dem:gridPropertyType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationProperty OfRasterRelief</td>
<td>xs:anyType</td>
<td>0–4</td>
</tr>
<tr>
<td><strong>MassPointReliefType</strong></td>
<td>dem:MassPointReliefType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td></td>
<td>reliefPoints gml:MultiPointPropertyType</td>
<td>0–4</td>
<td>0–4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfMassPointRelief</td>
<td>xs:anyType</td>
<td>0–4</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>dem:BreakLineReliefType</td>
<td>0–4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ridgeOrValleyLines</td>
<td>gml:MultiCurvePropertyType</td>
<td>0–4</td>
<td></td>
</tr>
<tr>
<td>breaklines</td>
<td>gml:MultiCurvePropertyType</td>
<td>0–4</td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfBreakLineRelief</td>
<td>xs:anyType</td>
<td>0–4</td>
<td></td>
</tr>
</tbody>
</table>

**CityFurniture module**

<table>
<thead>
<tr>
<th>frn:CityFurnitureType</th>
<th>1–4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
</tbody>
</table>

**CityObjectGroup module**

<table>
<thead>
<tr>
<th>grp:CityObjectGroupType</th>
<th>0–4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>groupMember</td>
<td>grp:CityObjectGroupMemberType</td>
</tr>
<tr>
<td>parent</td>
<td>grp:CityObjectGroupParentType</td>
</tr>
<tr>
<td>geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfCityObjectGroup</td>
<td>xs:anyType</td>
</tr>
</tbody>
</table>

**Generics module**

<table>
<thead>
<tr>
<th>gen:GenericCityObjectType</th>
<th>0–4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>lod0Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
</tr>
<tr>
<td>lod0TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
</tr>
<tr>
<td>lod0ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
</tr>
</tbody>
</table>

**LandUse module**

<table>
<thead>
<tr>
<th>luse:LandUseType</th>
<th>0–4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>lod0MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
</tr>
<tr>
<td>LOD</td>
<td>MultiSurface</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
</tr>
<tr>
<td>0</td>
<td>lod0Network</td>
</tr>
<tr>
<td>1</td>
<td>lod1MultiSurface</td>
</tr>
<tr>
<td>2</td>
<td>lod2MultiSurface</td>
</tr>
<tr>
<td>3</td>
<td>lod3MultiSurface</td>
</tr>
<tr>
<td>4</td>
<td>lod4MultiSurface</td>
</tr>
</tbody>
</table>

**Transportation module**

<table>
<thead>
<tr>
<th>XPath</th>
<th>xs:anyType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tran:AbstractTransportationObject</td>
<td>_GenericApplicationProperty OfTransportationObject</td>
<td>0 – 4</td>
</tr>
<tr>
<td>tran:TransportationComplexType</td>
<td>_GenericApplicationProperty OfTransportationObject</td>
<td>0 – 4</td>
</tr>
<tr>
<td>tran:TrafficAreaType</td>
<td>_GenericApplicationProperty OfTransportationComplex</td>
<td>1 – 4</td>
</tr>
<tr>
<td>tran:AuxiliaryTrafficAreaType</td>
<td>_GenericApplicationProperty OfTransportationComplex</td>
<td>1 – 4</td>
</tr>
<tr>
<td>tran:TrackType</td>
<td>_GenericApplicationProperty OfAuxiliaryTrafficArea</td>
<td>1 – 4</td>
</tr>
<tr>
<td>tran:RoadType</td>
<td>_GenericApplicationProperty OfTrack</td>
<td>1 – 4</td>
</tr>
<tr>
<td>tran:RailwayType</td>
<td>_GenericApplicationProperty OfRoad</td>
<td>1 – 4</td>
</tr>
<tr>
<td>tran:SquareType</td>
<td>_GenericApplicationProperty OfSquare</td>
<td>1 – 4</td>
</tr>
</tbody>
</table>

**Tunnel module**

<table>
<thead>
<tr>
<th>XPath</th>
<th>xs:anyType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tun:AbstractTunnelType</td>
<td>_GenericApplicationProperty OfTunnel</td>
<td>1 – 4</td>
</tr>
</tbody>
</table>

 año de construcción: yearOfConstruction
 año de demolición: yearOfDemolition
<table>
<thead>
<tr>
<th>lod2Solid</th>
<th>gml:SolidPropertyType</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>outerTunnelInstallation</td>
<td>tun:TunnelInstallationPropertyType</td>
<td>2–4</td>
</tr>
<tr>
<td>interiorTunnelInstallation</td>
<td>tun:IntTunnelInstallationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>boundedBy</td>
<td>tun:BoundarySurfacePropertyType</td>
<td>2–4</td>
</tr>
<tr>
<td>lod3Solid</td>
<td>gml:SolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>gml:MultiCurvePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>interiorHollowSpace</td>
<td>tun:InteriorHollowSpacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>consistsOfTunnelPart</td>
<td>tun:TunnelPartPropertyType</td>
<td>1–4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfAbstractTunnel</td>
<td>xs:anyType</td>
<td>1–4</td>
</tr>
<tr>
<td>tun:TunnelType</td>
<td></td>
<td>1–4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnel</td>
<td>xs:anyType</td>
<td>1–4</td>
</tr>
<tr>
<td>tun:WallSurfaceType</td>
<td></td>
<td>2–4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfRoofSurface</td>
<td>xs:anyType</td>
<td>2–4</td>
</tr>
<tr>
<td>tun:AbstractBoundarySurfaceType</td>
<td></td>
<td>2–4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBoundarySurface</td>
<td>xs:anyType</td>
<td>2–4</td>
</tr>
<tr>
<td>Class</td>
<td>xs:complexType</td>
<td>minOccurs</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>tun:GroundSurfaceType</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfGroundSurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>tun:ClosureSurfaceType</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfClosureSurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>tun:FloorSurfaceType</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfFloorSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>tun:InteriorWallSurfaceType</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfInteriorWallSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>tun:CeilingSurfaceType</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfCeilingSurface</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>tun:AbstractOpeningType</td>
<td></td>
<td>3 – 4</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOpening</td>
<td>xs:anyType</td>
<td>3 – 4</td>
</tr>
<tr>
<td>tun:WindowType</td>
<td>xs:anyType</td>
<td>3 – 4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfWindow</td>
<td>xs:anyType</td>
<td>3 – 4</td>
</tr>
<tr>
<td>tun:DoorType</td>
<td></td>
<td>3 – 4</td>
</tr>
<tr>
<td>address</td>
<td>core:AddressPropertyType</td>
<td>3 – 4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfDoor</td>
<td>xs:anyType</td>
<td>3 – 4</td>
</tr>
<tr>
<td>tun:HollowSpaceType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>gml:SolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>interiorFurniture</td>
<td>tun:InteriorFurniturePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>boundedBy</td>
<td>tun:BoundarySurfacePropertyType</td>
<td>4</td>
</tr>
<tr>
<td>hollowSpaceInstallation</td>
<td>tun:InfTunnelInstallationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfHollowSpace</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td>tun:TunnelFurnitureType</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>4</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>gml:GeometryPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnelFurniture</td>
<td>xs:anyType</td>
<td>4</td>
</tr>
<tr>
<td><strong>Vegetation module</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:AbstractVegetationObjectType</td>
<td></td>
<td>1 – 4</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfVegetationObject</td>
<td>xs:anyType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>veg:PlantCoverType</td>
<td></td>
<td>1 – 4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>averageHeight</td>
<td>gml:LengthType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>gml:MultiSurfacePropertyType</td>
<td>4</td>
</tr>
<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>--------------------------</td>
<td>---------------------------</td>
<td>---</td>
</tr>
<tr>
<td>lod3MultiSolid</td>
<td>gml:MultiSolidPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>Lod4MultiSolid</td>
<td>gml:MultiSolidPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfPlantCover</td>
<td>xs:anyType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>veg:SolitaryVegetationObjectType</td>
<td></td>
<td>1 – 4</td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>species</td>
<td>gml:CodeType</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>core:ImplicitRepresentationPropertyType</td>
<td>1</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>2</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>3</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>core:ImplicitRepresentationPropertyType</td>
<td>4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfSolitaryVegetationObject</td>
<td>xs:anyType</td>
<td>1 – 4</td>
</tr>
<tr>
<td>WaterObject module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wr:AbstractWaterObjectType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterObject</td>
<td>xs:anyType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>wr:WaterBodyType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>gml:CodeType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>function</td>
<td>gml:CodeType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>usage</td>
<td>gml:CodeType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>lod0MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>0</td>
</tr>
<tr>
<td>lod1MultiCurve</td>
<td>gml:MultiCurvePropertyType</td>
<td>1</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>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>wr:BoundedByWaterSurfacePropertyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterBody</td>
<td>xs:anyType</td>
<td>0 – 4</td>
</tr>
<tr>
<td>wr:AbstractWaterBoundarySurfaceType</td>
<td></td>
<td></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>_GenericApplicationProperty OfWaterBoundarySurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>wr:WaterSurfaceType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>waterLevel</td>
<td>WaterLevelType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterSurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>wr:WaterGroundSurfaceType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterGroundSurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
<tr>
<td>wr:WaterClosureSurfaceType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationProperty OfWaterClosureSurface</td>
<td>xs:anyType</td>
<td>2 – 4</td>
</tr>
</tbody>
</table>
Annex F
(informative)

Changelog for CityGML 2.0

The following table lists all feature types, properties, and data types which have been added or changed for CityGML 2.0. In order to maintain backwards compatibility, none of the classes marked as being deprecated in CityGML 1.0 has been deleted.

<table>
<thead>
<tr>
<th>Feature Class / Data Type</th>
<th>Property</th>
<th>New</th>
<th>Changed</th>
<th>Deleted</th>
<th>Description of change</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CityGML Core module</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>core:AbstractCityObjectType</td>
<td>relativeToTerrain</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>relativeToWater</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td><strong>Appearance module</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>app:AbstractTextureType</td>
<td>mimeType</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td><strong>Building module</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:AbstractBuildingType</td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>roofType</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>lod0FootPrint</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>lod0RoofEdge</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td>bldg:BuildingClassType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td>bldg:BuildingFunctionType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td>bldg:BuildingUsageType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td>bldg:RoofTypeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td>bldg:BuildingInstallationType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>X</td>
<td></td>
<td></td>
<td>replaced by gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>lod2ImplicitRepresentation</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>lod3ImplicitRepresentation</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>boundedBy</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td>bldg:BuildingInstallationClassType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingInstallationFunctionType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingInstallationUsageType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:IntBuildingInstallationType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>X</td>
<td></td>
<td></td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td></td>
<td>boundedBy</td>
<td>X</td>
<td></td>
<td></td>
<td>new attribute</td>
</tr>
<tr>
<td>bldg:IntBuildingInstallationClassType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:IntBuildingInstallation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FunctionType</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:IntBuildingInstallation</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UsageType</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:OuterCeilingSurfaceType</td>
<td>x</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterCeilingSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:OuterFloorSurfaceType</td>
<td>x</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterFloorSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:AbstractOpeningType</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:RoomType</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:RoomClassType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:RoomFunctionType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:RoomUsageType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingFurnitureType</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingFurnitureClassType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingFurnitureFunctionType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bldg:BuildingFurnitureUsageType</td>
<td>x</td>
<td>replaced by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gml:CodeType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bridge module</td>
<td>x</td>
<td>New module</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:AbstractBridgeType</td>
<td>x</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yearOfConstruction</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yearOfDemolition</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isMovable</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1Solid</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1MultiSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2Solid</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2MultiCurve</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>outerBridgeConstruction</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>outerBridgeInstallation</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interiorBridgeInstallation</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boundedBy</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3Solid</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3MultiCurve</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4Solid</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4MultiCurve</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>interiorBridgeRoom</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consistsOfBridgePart</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfAbstractBridge</td>
<td>x</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgeType</td>
<td>x</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Feature Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgePartType</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBridgePart</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgeConstructionElementType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1TerrainIntersection</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2TerrainIntersection</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3TerrainIntersection</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4TerrainIntersection</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod1ImplicitRepresentation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boundedBy</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBridgeConstructionElement</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgeInstallationType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boundedBy</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBridgeInstallation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:IntBridgeInstallationType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usage</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boundedBy</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfIntBridgeInstallation</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:AbstractBoundarySurfaceType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>opening</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBoundarySurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:RoofSurfaceType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfRoofSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:WallSurfaceType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfWallSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:OuterCeilingSurfaceType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterCeilingSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:OuterFloorSurfaceType</td>
<td>new feature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterFloorSurface</td>
<td>new attribute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class/Type</td>
<td>Feature/Property</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------</td>
<td>-------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OuterFloorSurface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:GroundSurfaceType</td>
<td>_GenericApplicationPropertyOf GroundSurface</td>
<td>x new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:ClosureSurfaceType</td>
<td>_GenericApplicationPropertyOf ClosureSurface</td>
<td>x new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:FloorSurfaceType</td>
<td>_GenericApplicationPropertyOf FloorSurface</td>
<td>x new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:InteriorWallSurfaceType</td>
<td>_GenericApplicationPropertyOf InteriorWallSurface</td>
<td>x new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:CeilingSurfaceType</td>
<td>_GenericApplicationPropertyOf CeilingSurface</td>
<td>x new feature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:AbstractOpeningType</td>
<td>lod3MultiSurface</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod3ImplicitRepresentation</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfOpening</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:WindowType</td>
<td>_GenericApplicationPropertyOfWindow</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:DoorType</td>
<td>address</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfDoor</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgeRoomType</td>
<td>class</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4Solid</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4MultiSurface</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>boundedBy</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>interiorFurniture</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bridgeRoomInstallation</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBridgeRoom</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>brid:BridgeFurnitureType</td>
<td>class</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4Geometry</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>lod4ImplicitRepresentation</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>_GenericApplicationPropertyOfBridgeFurniture</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CityFurniture module</td>
<td>frn:CityFurnitureType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>x new attribute</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>frn:CityFurnitureClassType</td>
<td>replaced by gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>frn:CityFurnitureFunctionType</td>
<td>replaced by gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CityObjectGroup module</td>
<td>grp:CityObjectGroupType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generics module</td>
<td>gen:GenericCityObjectType</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>class</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>function</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>usage</td>
<td>x gml:CodeType</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### LandUse module

| gen:GenericAttributeSetType | x | new data type |
| gen:Measure AttributeType | x | new data type |

#### luse:LandUseType

| class | x | gml:CodeType |
| function | x | gml:CodeType |
| usage | x | gml:CodeType |

#### luse:LandUseClassType

| usage | x | replaced by gml:CodeType |

#### luse:LandUseFunctionType

| usage | x | replaced by gml:CodeType |

#### luse:LandUseUsageType

| usage | x | replaced by gml:CodeType |

### Transportation module

#### tran:TransportationComplex Type

| class | x | new attribute |
| function | x | gml:CodeType |
| usage | x | gml:CodeType |

#### tran:TransportationComplex FunctionType

| usage | x | replaced by gml:CodeType |

#### tran:TransportationComplex UsageType

| usage | x | replaced by gml:CodeType |

#### tran:TrafficAreaType

| class | x | new attribute |
| function | x | gml:CodeType |
| usage | x | gml:CodeType |
| surfaceMaterial | x | gml:CodeType |

#### tran:TrafficAreaFunctionType

| surfaceMaterial | x | replaced by gml:CodeType |

#### tran:TrafficAreaUsageType

| surfaceMaterial | x | replaced by gml:CodeType |

#### tran:AuxiliaryTrafficAreaType

| class | x | new attribute |
| function | x | gml:CodeType |
| usage | x | new attribute |
| surfaceMaterial | x | gml:CodeType |

#### tran:AuxiliaryTrafficArea FunctionType

| surfaceMaterial | x | replaced by gml:CodeType |

### Tunnel module

#### tun:AbstractTunnelType

| class | x | new feature |
| function | x | new feature |
| usage | x | new feature |

#### tun:AbstractTunnelFunction

| yearOfConstruction | x | new attribute |
| yearOfDemolition | x | new attribute |

#### tun:AbstractTunnelUsage

<p>| lod1Solid | x | new attribute |
| lod1MultiSurface | x | new attribute |
| lod1TerrainIntersection | x | new attribute |
| lod2Solid | x | new attribute |
| lod2MultiSurface | x | new attribute |
| lod2MultiCurve | x | new attribute |
| lod2TerrainIntersection | x | new attribute |
| outerTunnelInstallation | x | new attribute |
| interiorTunnelInstallation | x | new attribute |
| boundedBy | x | new attribute |
| lod3Solid | x | new attribute |
| lod3MultiSurface | x | new attribute |
| lod3MultiCurve | x | new attribute |
| lod3TerrainIntersection | x | new attribute |
| lod4Solid | x | new attribute |
| lod4MultiSurface | x | new attribute |</p>
<table>
<thead>
<tr>
<th>Class/Feature</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>lod4TerrainIntersection</em></td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>interiorHollowSpace</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>consistsOfTunnelPart</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfAbstractTunnel</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:TunnelType</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnel</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:TunnelPartType</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnelPart</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:TunnelInstallationType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod3Geometry</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod2ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>boundedBy</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnelInstallation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:IntTunnelInstallationType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod2Geometry</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>boundedBy</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfIntTunnelInstallation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:AbstractBoundarySurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>lod2MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>opening</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfBoundarySurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:RoofSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfRoofSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:WallSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfWallSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:OuterCeilingSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterCeilingSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:OuterFloorSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOuterFloorSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:GroundSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfGroundSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:ClosureSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfClosureSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:FloorSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfFloorSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:InteriorWallSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfInteriorWallSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:CeilingSurfaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>Class Type</td>
<td>New Attribute</td>
<td>Reason</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOf CeilingSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:AbstractOpeningType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>lod3MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod3ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfOpening</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:WindowType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfWindow</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:DoorType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfDoor</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>tun:HollowSpaceType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4Solid</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4MultiSurface</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>boundedBy</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>interiorFurniture</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>hollowSpaceInstallation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfHollowSpace</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>tun:TunnelFurnitureType</td>
<td>x</td>
<td>new feature</td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4Geometry</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4ImplicitRepresentation</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>_GenericApplicationPropertyOfTunnelFurniture</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>Vegetation module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:PlantCoverType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>lod4MultiSolid</td>
<td>x</td>
<td>New Representation</td>
</tr>
<tr>
<td>veg:PlantCoverType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:PlantCoverClassType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>veg:PlantCoverFunctionType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:PlantCoverFunctionType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:SolitaireVegetationObject</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>new attribute</td>
</tr>
<tr>
<td>species</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>veg:PlantClassType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:PlantFunctionType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veg:SpeciesType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WaterObject module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wtr:WaterBodyType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>function</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>usage</td>
<td>x</td>
<td>gml:CodeType</td>
</tr>
<tr>
<td>wtr:WaterBodyClassType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wtr:WaterBodyFunctionType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wtr:WaterBodyUsageType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wtr:WaterSurfaceType</td>
<td></td>
<td></td>
</tr>
<tr>
<td>waterLevel</td>
<td>X</td>
<td>→ gml:CodeType</td>
</tr>
<tr>
<td>------------</td>
<td>---</td>
<td>----------------</td>
</tr>
<tr>
<td>wtr:WaterLevelType</td>
<td>X</td>
<td>replaced by gml:CodeType</td>
</tr>
</tbody>
</table>
Annex G
(informative)

Example CityGML datasets

The following chapters provide example CityGML models and excerpts of the corresponding CityGML instance documents. In chapters G.1 to G.6, a single CityGML building is subsequently illustrated and refined from a coarse LOD0 representation up to a semantically rich and geometric-topologically sound LOD4 model including the building interior. The model exemplifies concepts from the CityGML Building module and demonstrates the coherent modelling of semantics and geometric/topological properties. In chapter G.7, the building and its surrounding terrain are enriched with visual appearance information to show the usage of the CityGML Appearance module. A further example for the Appearance module is presented in chapter G.8 illustrating the texturing of polygons with inner holes. Finally, the use of a local engineering coordinate reference system in CityGML is illustrated in chapter G.9.

All CityGML instance documents contained in this annex are bundled with the CityGML schema package, and are available at http://schemas.opengis.net/citygml/examples/2.0/.

G.1 Example of a CityGML dataset for a building in LOD0

The first example illustrates a CityGML building in LOD0 surrounded by a terrain model represented as LOD0 Triangulated Irregular Network (dem:TINRelief). Both a footprint and a roof edge representation is assigned to the building (bldg:lod0FootPrint and bldg:lod0RoofEdge). The geometry is given as gml:MultiSurface following the conformance requirement no. 2 of the Building module (cf. chapter 10.3.9) according to which the height values of all 3D coordinate tuples belonging to the same surface must be identical (which results in horizontal surfaces). The footprint (painted in dark blue) is located at the lowest elevation of the building whereas the roof edge representation (painted in light cyan) is at roof level (eaves height) and hence is floating over the terrain. The building is assigned address information using the xAL address schema.

Fig. 75: Example of a CityGML building model (footprint and roof edge) and a terrain model in LOD0 (left: 3D graphic; right: model hierarchy). The footprint (painted in dark blue) is located at the lowest elevation of the building whereas the roof edge representation (painted in light cyan) is at roof level (eaves height) and hence floating over the terrain.

<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.opengis.net/citygml/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
  xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
  xmlns:xAL="urn:oasis:names:tc:ciq:xsd:schema:xAL:2.0"
  xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd"/>

Copyright © 2012 Open Geospatial Consortium.
<gml:name>Simple 3D city model LOD0 without Appearance</gml:name>
<gml:boundedBy>
  <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs:crs:EPSG::25832,crs:EPSG::5783">
    <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
    <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
    <gml:name>Example Building LOD0</gml:name>
    <bldg:function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
    <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
    <bldg:roofType codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
    <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
    <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
    <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
  </bldg:Building>
</cityObjectMember>
<bldg:lod0FootPrint>
  <gml:MultiSurface>
    <gml:Polygon>
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438350.0 112.0</gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:MultiSurface>
</bldg:lod0FootPrint>
<bldg:lod0RoofEdge>
  <gml:MultiSurface>
    <gml:Polygon>
      <gml:exterior>
        <gml:LinearRing>
          <gml:posList>458874.6 5438355.312347524 115.0 458874.6 5438349.687652476 115.0 458885.4 5438349.687652476 115.0 458885.4 5438355.312347524 115.0 458874.6 5438355.312347524 115.0</gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:MultiSurface>
</bldg:lod0RoofEdge>
<bldg:address>
  <xalAddress>
    <xAL:AddressDetails>
      <xAL:Country>Germany</xAL:Country>
      <xAL:CountryName>Germany</xAL:CountryName>
      <xAL:Locality Type="Town">
        <xAL:LocalityName>Eggenstein-Leopoldshafen</xAL:LocalityName>
        <xAL:ThoroughfareTypes="Street">
          <xAL:ThoroughfareNumber>1</xAL:ThoroughfareNumber>
          <xAL:ThoroughfareName>Hermann-von-Helmholtz-Platz</xAL:ThoroughfareName>
        </xAL:Thoroughfare>
        <xAL:PostalCode>76344</xAL:PostalCode>
      </xAL:Locality>
    </xAL:AddressDetails>
  </xalAddress>
</bldg:address>
Listing 3: Excerpt from the CityGML dataset for a building in LOD0 visualised in Fig. 75.
G.2 Example of a CityGML dataset for a building in LOD1

The following Fig. 76 shows both the building and the terrain model in LOD1. The building is represented as block model. Its geometry is described by a solid (gml:Solid) whose exterior shell is bounded by six surfaces (gml:Polygon). In addition to the geometry, the building is enriched with further thematic attributes. Some of these attributes are enumerative and demonstrate the usage of the CityGML code list mechanism (cf. chapter 10.14). The coded attribute values are taken from the code lists proposed by the SIG 3D (cf. Annex C).

<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.opengis.net/citygml/2.0"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
    xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
    xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
    xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd"
    http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd">
    <gml:name>Simple 3D city model LOD1 without Appearance</gml:name>
    <gml:boundedBy>
        <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
            <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
            <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <cityObjectMember>
        <bldg:Building gml:id="GML_7b1a5a6fd-dad-4c3d-a307-3eb9ee0a8e68">
            <gml:name>Example Building LOD1</gml:name>
            <bldg:function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
            <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
            <bldg:roofType codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
            <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
            <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
            <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
            <bldg:lod1Solid>
                <gml:Solid>
                    <gml:CompositeSurface>
                        <!-- Face Side 1 -->
                        <gml:surfaceMember>
                            <gml:Polygon>
                                <gml:exterior>
                                    <gml:posList>458875.0 5438350.0 112.0 458885.0 5438350.0 112.0 458885.0 5438350.0 116.0 458875.0 5438350.0 116.0 458875.0 5438350.0 112.0</gml:posList>
                                </gml:exterior>
                            </gml:Polygon>
                        </gml:surfaceMember>
                        <!-- Face Side 2 -->
                    </gml:CompositeSurface>
                </gml:Solid>
            </bldg:lod1Solid>
        </bldg:Building>
    </cityObjectMember>
</CityModel>
<gml:surfaceMember>
  <gml:Polygon>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>
          458885.0 5438350.0 112.0 458885.0 5438355.0 112.0 458885.0 5438355.0 116.0 458885.0 5438350.0 116.0 458885.0 5438355.0 112.0
        </gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</gml:surfaceMember>

<!-- Face Side 3 -->
<gml:surfaceMember>
  <gml:Polygon>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>
          458885.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438355.0 116.0 458885.0 5438355.0 116.0 458885.0 5438355.0 112.0
        </gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</gml:surfaceMember>

<!-- Face Side 4 -->
<gml:surfaceMember>
  <gml:Polygon>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>
          458875.0 5438350.0 116.0 458875.0 5438350.0 116.0 458875.0 5438355.0 116.0 458875.0 5438355.0 116.0 458875.0 5438350.0 116.0
        </gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</gml:surfaceMember>

<!-- Face Top -->
<gml:surfaceMember>
  <gml:Polygon>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>
          458875.0 5438350.0 116.0 458875.0 5438355.0 116.0 458875.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438350.0 112.0
        </gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</gml:surfaceMember>

<!-- Face Bottom -->
<gml:surfaceMember>
  <gml:Polygon>
    <gml:exterior>
      <gml:LinearRing>
        <gml:posList>
          458875.0 5438350.0 112.0 458875.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438355.0 112.0 458875.0 5438350.0 112.0
        </gml:posList>
      </gml:LinearRing>
    </gml:exterior>
  </gml:Polygon>
</gml:surfaceMember>

<xAL:AddressDetails>
  <xAL:Address>
    <xAL:Country>Germany</xAL:Country>
    <xAL:Locality>Germany</xAL:Locality>
    <xAL:PostalCode>76344</xAL:PostalCode>
  </xAL:Address>
</xAL:AddressDetails>
Listing 4: Excerpt from the CityGML dataset for a building in LOD1 visualised in Fig. 76.
G.3 Example of a CityGML dataset for a building in LOD2

The building model from the previous chapters is now represented in LOD2. The model reflects the actual roof structure and contains boundary surfaces (bldg:boundedBy) which semantically classify the surfaces of the exterior building shell (bldg:RoofSurface, bldg:WallSurface, and bldg:GroundSurface). In addition to the thematic boundary surfaces, the building geometry is also described using an LOD2 solid geometry (gml:Solid). According to conformance requirement no. 4 of the Building module (cf. chapter 10.3.9), this solid geometry must reference the geometry of the boundary surfaces by using the GML3 XLink mechanism (xlink:href, printed in bold in the following Listing 5).

<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/citygml/2.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd">
<gml:name>Simple 3D city model LOD2 without Appearance</gml:name>
<gml:boundedBy>
  <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs:crs:EPSG::25832,crs:EPSG::5783">
    <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
    <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
    <gml:name>Example Building LOD2</gml:name>
    … (further attributes see LOD1 example)
  </bldg:Building>
</cityObjectMember>
</CityModel>

Fig. 77: Example of a CityGML building model in LOD2 (left: 3D graphic; right: model hierarchy).
Listing 5: Excerpt from the CityGML dataset for a building in LOD2 visualised in Fig. 77.
G.4 Example of a CityGML dataset for a building in LOD2 with an adjacent building part illustrating CityGML’s topology representation

This example illustrates CityGML’s topology representation which uses the XLink mechanism of GML3 (cf. chapter 8.1). The LOD2 model of annex G.3 is extended by placing an adjacent garage next to the building (cf. Fig. 78). The garage is modelled as building part (bldg:BuildingPart) and shares a common surface geometry with the building shell. For both the building and the garage thematic boundary surfaces (bldg:boundedBy) as well as separate solid geometries are given. The wall surface of the building where the garage touches the building is depicted in Fig. 79. For the building, the surface geometry is split into a non-shared and a shared part (see bldg:WallSurface with gml:name “Wall East”). The latter is referenced by the garage both by its corresponding bldg:WallSurface (gml:name “Garage Wall West”) and its solid geometry. Since the orientation of the shared surface has to be reversed in the context of the garage, the XLink reference is embraced by a gml:OrientableSurface element. The XLinks pointing from the garage to the building geometry explicitly denote the topological adjacency relation between both features.

Fig. 78: Example of a CityGML building model in LOD2 with an adjacent building part (garage) which shares a common boundary surface (left: 3D graphic; right: model hierarchy).

Fig. 79: The boundary surface where the garage touches the building. For the building, the geometry is split into a non-shared part (“Wall East”, Polygon 1) and a shared part (“Wall East”, Polygon 2). Only the latter is referenced by the garage using the XLink mechanism.

<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.opengis.net/citygml/2.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
schemaLocation="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd"
http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd">
<gml:name>3D city model LOD2 without Appearance</gml:name>
<gml:boundedBy>
<gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs:EPSG::25832,crs:EPSG::5783">
<gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
<gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<cityObjectMember>

WallSurface “Wall East”, Polygon 1
WallSurface “Wall East”, Polygon 2
WallSurface “Garage Wall West” (XLink to “Wall East”, Polygon 2)

</cityObjectMember>
</CityModel>
<bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
  <gml:name>Example Building LOD2</gml:name>
  ... (further attributes see LOD1 example)
  <bldg:lod2Solid>
    <gml:Solid>
      <gml:exterior>
        <gml:CompositeSurface>
          <gml:surfaceMember xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757"/>
          <!-- Ground Slab -->
          <gml:surfaceMember xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1"/>
          <!-- Wall North -->
          <gml:surfaceMember xlink:href="#GML_d3990900-2f18-4472-8886-1c127e67d1f1"/>
          <!-- Wall East 1 -->
          <gml:surfaceMember xlink:href="#GML_56d1dd88-36dd-4d1e-bff0-3305f6fa778"/>
          <!-- Wall East 2 -->
          <gml:surfaceMember xlink:href="#GML_9f0465e6-f316-4f89-a9bd-eb21934ffe84"/>
          <!-- Wall West -->
          <gml:surfaceMember xlink:href="#GML_5cc4fd92-d5de-4d88-971e-892e91da2d9f"/>
          <!-- Roof North -->
          <gml:surfaceMember xlink:href="#GML_ec6a8966-58d9-4894-8ecd-9ac8b9b9231f"/>
          <!-- Roof South -->
          <gml:surfaceMember xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583d05e3"/>
        </gml:CompositeSurface>
      </gml:exterior>
    </gml:Solid>
  </bldg:lod2Solid>
</bldg:Building>

Copyright © 2012 Open Geospatial Consortium.
Listing 6: Excerpt from the CityGML dataset for a building in LOD2 having an adjacent garage as visualised in Fig. 78.
G.5 Example of a CityGML dataset for a building in LOD3

The LOD3 building model illustrated in this chapter (cf. Fig. 80) adds doors (bldg:Door), windows (bldg:Window) and roof overhangs (bldg:RoofSurface) to the LOD2 representation of the previous chapters. Again, a solid geometry for the LOD3 of the building is realized by referencing the geometries of the thematic boundary surfaces using the GML3 XLink mechanism (xlink:href). In order to get a valid solid geometry, the roof surfaces are geometrically split into the roof slab and the roof overhang parts. Only the geometry of the roof slab is referenced by the solid. Walls composed of several surfaces (e.g. reveals) are modeled as gml:CompositeSurface which then is referenced by the building solid. Boundary surfaces containing openings for doors or windows are modeled with polygons having one exterior and several interior linear rings (according to conformance requirement no. 8 of the Building module, cf. chapter 10.4.8).

Fig. 80: Example of a CityGML building model in LOD3 (left: 3D graphic; right: model hierarchy).
Listing 7: Excerpt from the CityGML dataset for the buildings in LOD3 visualised in Fig. 80.
G.6 Example of a CityGML dataset for a building in LOD4

In LOD4, the building is completed with the representation of the building interior (cf. Fig. 81). The model contains a room (bldg:Room), which is equipped with a rocking chair (bldg:BuildingFurniture). The room is bounded by interior boundary surfaces (bldg:InteriorWallSurface, bldg:FloorSurface, bldg:CeilingSurface, associated through the bldg:boundedBy property of the room) whose geometries are referenced by the LOD4 solid geometry of the room (xlink:href). If the normal vector of an interior boundary surface is pointing into the room, its orientation has to be flipped using an orientable surface (gml:OrientableSurface) when referenced from the solid in order to create a valid solid geometry (for a gml:Solid, the normal vectors of the surfaces bounding the volume have to point outwards the volume).

Fig. 81: Example of a CityGML building model in LOD4. The roof surfaces are not shown in order to visualize the interior boundary surfaces as well as the building furniture (left: 3D graphic; right: model hierarchy).
Listing 8: Excerpt from the CityGML dataset for the buildings in LOD4 visualised in Fig. 81.

```xml
<Building>
  ...
  <address>...
    (address see LOD1 example)
  </address>
  <AddressReference/>
  <BuildingReference/>
  ...
  <BuildingFurniture>
    <BuildingFurniture>
      <name>Rocking Chair</name>
      <function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/BuildingFurniture_function.xml">1340</function>
      ...
      <MultiSurface>
        ...
      </MultiSurface>
      ...<MultiSurface>
      ...
      <lod4MultiSurface>
        ...
      </lod4MultiSurface>
      ...
    </BuildingFurniture>
  </BuildingFurniture>
  ...
  <Room/>
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
  ...
```

Listing 8: Excerpt from the CityGML dataset for the buildings in LOD4 visualised in Fig. 81.
G.7 Example of a CityGML dataset illustrating the appearance model

The following CityGML dataset is based on the simple building model from chapters G.2 and G.3 given in LOD1 and LOD2. Furthermore two separate appearance themes are defined – a summer theme and a winter theme – describing different visual appearances for the building and the surrounding terrain. Each LOD has an individual appearance for these specific themes.

Several concepts of CityGML’s appearance model are used in this dataset. Regarding LOD1, an X3DMaterial object defines the material of the whole building which is applied to all of its surfaces. In addition, a GeoreferencedTexture is assigned both to the terrain and the roof surface of the building. In LOD2 the vertical surfaces of the building are texturised individually using ParameterizedTexture objects whereas the roof surfaces and the terrain again are described by a GeoreferencedTexture. The texture mapping for the GeoreferencedTexture objects is given inline (cf. Listing 9) as well as using an ESRI world file (cf. Listing 11). The modelling approach results in four possible visualizations of the dataset that are represented in Fig. 82 and Fig. 83.

![Fig. 82: Visualisation of a simple building in LOD1 using CityGML’s appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer and (b) showing the building in winter (image: Hasso-Plattner-Institute).](image1)

![Fig. 83: Visualisation of a simple building in LOD2 using CityGML’s appearance model. Two themes are defined for the building and the surrounding terrain: (a) theme showing the building in summer and (b) showing the building in winter (image: Hasso-Plattner-Institute).](image2)

<?xml version="1.0" encoding="utf-8"?>
<CityModel
xmlns="http://www.opengis.net/citygml/2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:app="http://www.opengis.net/citygml/appearance/2.0"
xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<gml:boundedBy>
  <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs:EPSG::25832,crs:EPSG::5783">
    <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
    <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
  <bldg:Building gml:id="GML_7b1a506f-ddad-4c3d-a507-3eb9ee0a868">
    <bldg:function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
    <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
    <bldg:roofType codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
    <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
    <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
    <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
  </bldg:Building>
</cityObjectMember>
<gml:surfaceMember>
  <gml:surfaceMember>
    <gml:Polygon gml:id="fFront">
      <gml:exterior>
        <gml:LinearRing gml:id="fFrontExt">
          <gml:posList srsDimension="3">
            458885 5438350 112 458885 5438355 112 458885 5438355 115 458885 5438352.5 117 458885 5438350 115 458885 5438350 112
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:surfaceMember>
  <gml:surfaceMember>
    <gml:Polygon gml:id="fRight">
      <gml:exterior>
        <gml:LinearRing gml:id="fRightExt">
          <gml:posList srsDimension="3">
            458875 5438355 112 458875 5438355 115 458885 5438355 115 458885 5438352.5 115 458885 5438355 112
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:surfaceMember>
  <gml:surfaceMember>
    <gml:Polygon gml:id="fBack">
      <gml:exterior>
        <gml:LinearRing gml:id="fBackExt">
          <gml:posList srsDimension="3">
            458875 5438350 115 458885 5438350 115 458885 5438355 117 458875 5438352.5 117 458875 5438355 115 458875 5438355 112
          </gml:posList>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:surfaceMember>
</gml:Polygon>
</gml:surfaceMember>
</gml:CompositeSurface>
</gml:exterior>
</gml:Solid>
</bldg:lod2Solid>
</bldg:Building>
</cityObjectMember>
</cityObjectMember>
<dem:ReliefFeature gml:id="GML_6bb30328-7599-4500-90ef-766def6aa67b">
  <dem:lod>1</dem:lod>
  <dem:reliefComponent>
    <dem:TINRelief gml:id="GML_4eb161b0-aa7e-4087-937c-5c4c427c7fc9">
      <gml:name>Ground</gml:name>
    </dem:TINRelief>
  </dem:reliefComponent>
</dem:ReliefFeature>
<dem:lod>1</dem:lod>
<dem:tin>
<dem:TriangulatedSurface gml:id="ground">
<dem:trianglePatches>
<dem:Triangle>
<dem:exterior>
<dem:LinearRing>
<dem:posList>458868 5438362 112 458875 5438355 112 458883 5438362 114 458868 5438362 112 458870 5438360</dem:posList>
</dem:LinearRing>
</dem:exterior>
</dem:Triangle>
...
</dem:trianglePatches>
</dem:TriangulatedSurface>
</dem:tin>
</dem:TINRelief>
</dem:reliefComponent>
</dem:ReliefFeature>
</cityObjectMember>
</app:appearanceMember>

<app:Appearance>
<app:theme>Summer</app:theme>
<app:surfaceDataMember>
<app:X3DMaterial gml:id="lod1Material">
<app:diffuseColor>1.0 0.6 0.0</app:diffuseColor>
</app:X3DMaterial>
</app:surfaceDataMember>
</app:Appearance>

<app:GeoreferencedTexture>
<app:imageURI>ground_summer.png</app:imageURI>
<app:referencePoint>
<dem:Point>
<dem:pos srsDimension="2">458870 5438360</dem:pos>
</dem:Point>
</app:referencePoint>
<app:orientation>0.05 0.0 0.0 -0.05</app:orientation>
</app:GeoreferencedTexture>

<app:ParameterizedTexture gml:id="sideTexture">
<app:imageURI>facade.png</app:imageURI>
<app:TexCoordList gml:id="fLeft">
<app:textureCoordinates ring="#fLeftExt1">0.0 0.0 2.0 0.0 2.0 1.0 0.0 1.0 0.0 0.0</app:textureCoordinates>
<app:textureCoordinates ring="#fLeftExt2">2.0 0.0 4.0 0.0 4.0 1.0 2.0 1.0 2.0 0.0</app:textureCoordinates>
</app:TexCoordList>
</app:ParameterizedTexture>

<app:ParameterizedTexture gml:id="frontTexture">
<app:imageURI>front_back_summer.png</app:imageURI>
<app:TexCoordList gml:id="frontTexCoord">
<app:textureCoordinates ring="#fFrontExt">0.0 0.0 0.5 0.0 0.5 0.6 0.25 1.0 0.0 0.6 0.0
0.0</app:textureCoordinates>
</app:TexCoordList>
</app:ParameterizedTexture>

<app:ParameterizedTexture gml:id="backTexture">
<app:imageURI>back.png</app:imageURI>
<app:TexCoordList gml:id="backTexCoord">
<app:textureCoordinates ring="#fBackExt">0.5 0.0 1.0 0.0 1.0 0.6 0.75 1.0 0.5 0.6 0.5
0.0</app:textureCoordinates>
</app:TexCoordList>
</app:ParameterizedTexture>
Listing 9: Excerpt from the CityGML dataset illustrating CityGML’s appearance model. The dataset is visualised in Fig. 82 and Fig. 83.

The following three raster images (Fig. 84 - Fig. 86) are referenced in the dataset by ParameterizedTexture objects to texturize the vertical boundary surfaces of the building in LOD2. The image facade.png (cf. Fig. 84) is assigned to the side surfaces using the texture wrapping mode wrap and is applied both within the summer and the winter theme.

Fig. 84: Image facade.png used in the dataset to texturize the side surfaces of the building in LOD2 (cf. Fig. 83 a. and b.) (image: Hasso-Plattner-Institute).

Fig. 85 shows the texture atlas front_back_summer.png combining the textures for the front surface and the back surface of the building in LOD2 within the summer theme. Only a portion of this image is assigned to the specific surfaces. The relevant parts are defined using a TextCoordList object.
Fig. 85: Texture atlas `front_back_summer.png` containing the textures for the front surface and the back surface of the building in LOD2 within the summer theme (cf. Fig. 83 a.) (image: Hasso-Plattner-Institute).

Identically to `front_back_summer.png` the texture atlas `front_back_winter.png` contains the textures for the front surface and the back surface of the building in LOD2 within the winter theme.

Fig. 86: Texture atlas `front_back_winter.png` containing the textures for the front surface and the back surface of the building in LOD2 within the winter theme (cf. Fig. 83 b.) (image: Hasso-Plattner-Institute).

The raster images shown in Fig. 87 and Fig. 88 are assigned to the terrain and the roof surfaces of the building in LOD1 as well as in LOD2. In the dataset this is implemented by a `GeoreferencedTexture` object linking to the according GML geometry objects. Whereas the image `ground_summer.png` (cf. Fig. 87) represents the texture for the summer theme, `ground_winter.png` (cf. Fig. 88) is used within the winter theme.

Fig. 87: The image `ground_summer.png` is assigned to the terrain and the roof surfaces of the building both in LOD1 and LOD2 (cf. Fig. 82 a. and Fig. 83 a.) within the summer theme (image: Hasso-Plattner-Institute).
Instead of denoting the georeference of the GeoreferencedTexture elements inline as shown in Listing 9, the Appearance module also supports to provide the texture mapping with the image file (e.g. using a georeferenced TIFF) or as a separate ESRI world file. The latter is a six-line text file with a decimal number on each line. These numbers correspond to the parameters of the inline texture mapping and can be 1:1 copied from there (and vice versa). The rotation and scaling parameters given by the orientation property of GeoreferencedTexture (2x2 matrix represented as a list of 4 doubles in row-major) are given on lines 1, 3, 2, and 4 in an ESRI world file. And the location of the center of the upper left image pixel in world space denoted by the referencePoint property of GeoreferencedTexture corresponds to lines 5 and 6. For the example in Listing 9, the world files for both texture images contain the following values:

```
0.05
0.0
0.0
-0.05
458870
5438360
```

Listing 10: Contents of a world file specifying the same texture mapping for a GeoreferencedTexture as given by the inline georeference shown in Listing 9.

In order to make use of this world file, it has to be stored together with the texture image (for filename conventions, see http://en.wikipedia.org/wiki/World_file). Additionally, the inline texture mapping has to be removed from both GeoreferencedTexture elements in Listing 9 as shown in the following excerpt. Both alternatives for denoting the georeference of a GeoreferencedTexture are bundled with the CityGML schemas, and are available at http://schemas.opengis.net/citygml/examples/2.0/appearance/.

```
...<app:GeoreferencedTexture>
  <app:imageURI>ground_winter.png</app:imageURI>
  <app:target>ground</app:target>
  <app:target>#lod1RoofPoly1</app:target>
  <app:target>#lod2RoofPoly1</app:target>
  <app:target>#lod2RoofPoly2</app:target>
</app:GeoreferencedTexture>
...
```

Listing 11: Adapted GeoreferencedTexture element for which the texture mapping is provided by the world file shown in Listing 10.
G.8 Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes

The following example demonstrates the use of texture coordinates for complex surfaces with holes. Additionally, the concept of overwriting (cf. chapter 9.1) is exemplified. The example shows a textured road in LOD1, which consists of a roundabout and two entering streets (Fig. 89). It uses two `app:ParameterizedTexture` objects, a regular road piece (`rd`, Fig. 90) and a dirt track becoming a paved road (`dt`, Fig. 91). All geometry is contained in a single `gml:MultiSurface (road)` being the target of `rd`. The roundabout is modelled as a polygon with holes (`roundaboutPoly`). Both rings `raEx` and `raIn` need to receive texture coordinates. For efficient texturing, texture coordinates and wrap mode are chosen such that the regular road piece is draped onto the polygonal ring as desired at the price of some distortion. A distortion-free alternative, e.g. for higher LODs, requires an additional road piece texture for the roundabout (Fig. 92) or a texture for the complete roundabout (Fig. 93). In both cases, texture space is wasted. Wasted areas are marked red.

![Example of a CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes](image)

Fig. 89: A rendering of the textured geometry (images for this example: Hasso-Plattner-Institute).

![The road piece texture.](image)

Fig. 90: The road piece texture.

![The dirt track texture.](image)

Fig. 91: The dirt track texture.

![A distortion-free road piece for the roundabout. Red areas never become visible and are wasted.](image)

Fig. 92: A distortion-free road piece for the roundabout. Red areas never become visible and are wasted.
The entering dirt track requires a different texture. Even though its geometry (dirtPoly) is contained in the already textured road, the existing texture is overwritten and replaced by using dirtPoly as target of dt. Overwriting only occurs when assigning a new texture to a surface geometry object contained in an already textured aggregated geometry object. Assigning two textures to the same surface geometry object directly is not allowed. The same applies to materials.

```xml
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/2.0"
xmlns:app="http://www.opengis.net/citygml/appearance/2.0"
xmlns:tran="http://www.opengis.net/citygml/transportation/2.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/citygml/appearance/2.0 http://schemas.opengis.net/citygml/appearance/2.0/appearance.xsd
http://www.opengis.net/citygml/transportation/2.0 http://schemas.opengis.net/citygml/transportation/2.0/transportation.xsd"
>
<gml:boundedBy>
<!--The srsName attribute references a local engineering CRS that is defined in the example dataset provided in annex G.9 -->
<gml:Envelope srsDimension="3" srsName="local-CRS-1">
<gml:lowerCorner>-45.0 -20.0 0.0</gml:lowerCorner>
<gml:upperCorner>45.0 20.0 10.0</gml:upperCorner>
</gml:Envelope>
</gml:boundedBy>
<cityObjectMember>
<tran:Road>
<app:appearance>
<app:theme>visual</app:theme>
<app:surfaceDataMember>
<app:ParameterizedTexture gml:id="rd">
<app:imageURI>rd.png</app:imageURI>
<app:wrapMode>mirror</app:wrapMode>
<app:target uri="#road">
<app:TexCoordList>
<app:textureCoordinates ring="#raEx">0 1 1 1 0 1 1 1 0 1 1 1</app:textureCoordinates>
<app:textureCoordinates ring="#raIn">0 0 1 0 0 0 1 0 0 0 1 0</app:textureCoordinates>
<app:textureCoordinates ring="#roadEx">0 0 2.5</app:textureCoordinates>
<app:textureCoordinates ring="#dirtEx">0 0 2.5</app:textureCoordinates>
</app:TexCoordList>
</app:target>
</app:ParameterizedTexture>
</app:surfaceDataMember>
<app:surfaceDataMember>
<app:ParameterizedTexture gml:id="dt">
<app:imageURI>dt.png</app:imageURI>
<app:wrapMode>mirror</app:wrapMode>
<app:target uri="#dirtPoly">
<app:TexCoordList>
<app:textureCoordinates ring="#dirtEx">0 0 1 0 1 1 0 1 0 0</app:textureCoordinates>
</app:TexCoordList>
</app:target>
</app:ParameterizedTexture>
</app:surfaceDataMember>
</app:appearance>
<tran:lod1MultiSurface>
<gml:MultiSurface gml:id="road">
<gml:surfaceMember>
<app:ParameterizedTexture gml:id="rd">
<app:imageURI>rd.png</app:imageURI>
<app:wrapMode>mirror</app:wrapMode>
<app:target uri="#road">
<app:TexCoordList>
<app:textureCoordinates ring="#raEx">0 1 1 1 0 1 1 1 0 1 1 1</app:textureCoordinates>
<app:textureCoordinates ring="#raIn">0 0 1 0 0 0 1 0 0 0 1 0</app:textureCoordinates>
<app:textureCoordinates ring="#roadEx">0 0 2.5 0 2.5 1 0 1 0 1</app:textureCoordinates>
<app:textureCoordinates ring="#dirtEx">0 0 2.5 0 2.5 1 0 1 0</app:textureCoordinates>
</app:TexCoordList>
</app:target>
</app:ParameterizedTexture>
</gml:surfaceMember>
</gml:MultiSurface>
</tran:lod1MultiSurface>
</app:Appearance>
<app:appearance>
<app:theme>visual</app:theme>
<app:surfaceDataMember>
<app:ParameterizedTexture gml:id="dt">
<app:imageURI>dt.png</app:imageURI>
<app:wrapMode>mirror</app:wrapMode>
<app:target uri="#dirtPoly">
<app:TexCoordList>
<app:textureCoordinates ring="#dirtEx">0 0 1 0 1 0 1 0</app:textureCoordinates>
</app:TexCoordList>
</app:target>
</app:ParameterizedTexture>
</gml:surfaceDataMember>
</app:appearance>
</tran:Road>
</cityObjectMember>
</CityModel>
```
Listing 12: CityGML dataset illustrating the use of texture coordinates for complex surfaces with holes. The dataset is visualised in Fig. 89.
G.9  Example of a CityGML dataset illustrating the use of local coordinate reference systems

The following dataset demonstrates the use of local engineering coordinate reference systems in CityGML. The dataset is based on the LOD1 example given in annex G.2. The definition of the local CRS is encoded in the gml:metaDataProperty of the CityModel element and describes a 3D Cartesian reference system. Its origin corresponds to an anchor point which references a point on the earth’s surface (here: in Germany). This anchor point is defined in the EngineeringDatum element that is used by the local CRS, and is associated with a compound 3D CRS which combines a Projected CRS (ETRS89 / UTM zone 32N; EPSG code 25832) for the planimetry with a Vertical CRS (DHHN92 height, EPSG code 5783) for the height reference. The gml:Envelope of the CityModel references the local CRS by its gml:id “local-CRS-1” and thus it is inherited by all geometries in all properties of the CityModel and members of this feature collection. If required, the reference system may be overridden by the members of the CityModel, for example, to provide a separate anchor point for each city object.

```xml
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/citygml/2.0"
xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xml:gml="http://www.opengis.net/gml"
xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
xlink:href="http://www.opengis.net/citygml/building/2.0 http://schemas.opengis.net/citygml/building/2.0/building.xsd http://www.opengis.net/citygml/relief/2.0 http://schemas.opengis.net/citygml/relief/2.0/relief.xsd">
<gml:metaDataProperty>
<metaData:CommonMetaData>
<metadata:type>engineering</metadata:type>
</gml:metaDataProperty>
<gml:srsName codeSpace="XYZ">urn:ogc:def:crs:local:CRS:1</gml:srsName>
<gml:scope>CityGML</gml:scope>
<gml:usesCS>
<gml:CartesianCS gml:id="local-CS-1">
<gml:metaDataProperty>
<metaData:CommonMetaData>
<metadata:type>Cartesian</metadata:type>
</gml:metaDataProperty>
<gml:csName codeSpace="XYZ">urn:ogc:def:crs:local:CS:1</gml:csName>
<gml:usesAxis>
<gml:CoordinateSystemAxis gml:id="local-axis-1" gml:uom="urn:ogc:def:uom:EPSG::9001">
<gml:name/>
<gml:axisID>X</gml:axisID>
</gml:CoordinateSystemAxis>
<gml:CoordinateSystemAxis gml:id="local-axis-2" gml:uom="urn:ogc:def:uom:EPSG::9001">
<gml:name/>
<gml:axisID>Y</gml:axisID>
</gml:CoordinateSystemAxis>
</gml:usesAxis>
</gml:CartesianCS>
</gml:usesCS>
</gml:metaDataProperty>
</CityModel>
```
<gml:usesAxis>
  <gml:CoordinateSystemAxis gml:id="local-axis-3" gml:uom="urn:ogc:def:uom:EPSG::9001">
    <gml:name/>
    <gml:axisID>
      <gml:name>Z</gml:name>
      <gml:axisID>
        <gml:name>Z</gml:name>
      </gml:axisID>
      <gml:axisAbbrev>z</gml:axisAbbrev>
      <gml:axisDirection codeSpace="XYZ">Z</gml:axisDirection>
    </gml:CoordinateSystemAxis>
  </gml:usesAxis>
</gml:CartesianCS>

<gml:usesEngineeringDatum>
  <gml:EngineeringDatum gml:id="local-datum-1">
    <gml:metaDataProperty>
      <metadata:CommonMetaData>
        <metadata:type>Cartesian datum</metadata:type>
      </metadata:CommonMetaData>
    </gml:metaDataProperty>
    <gml:datumName codeSpace="XYZ">Datum1</gml:datumName>
    <gml:anchorPoint codeSpace="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">458868.0 5438343.0 112.0</gml:anchorPoint>
    <!-- The anchor point defines the origin of the local CS with respect to the world CRS -->
    <!-- In this example, the anchor point references a point on the earth (in Germany) using a compound CRS -->
    <!-- For planimetry, the reference system ETRS89 / UTM zone 32N (EPSG code 25832) is used -->
    <!-- The vertical reference system is DHHN92 height (EPSG code 5783) -->
  </gml:EngineeringDatum>
</gml:usesEngineeringDatum>

<gml:metaDataProperty>
  <gml:description>Simple example for a CityGML dataset using a local engineering CRS</gml:description>
  <gml:name>Simple 3D city model LOD1 without Appearance</gml:name>
</gml:metaDataProperty>

<gml:boundedBy>
  <gml:Envelope srsName="#local-CRS-1">
    <!-- Encoding of local-CRS-1 is specified in CityModel metaDataProperty in this document-->
    <gml:pos srsDimension="3">0.0 0.0 0.0</gml:pos>
    <gml:pos srsDimension="3">24.0 19.0 4.0</gml:pos>
  </gml:Envelope>
</gml:boundedBy>

<cityObjectMember>
  <bldg:Building gml:id="GML_7b1a5a6f-ddad-4cl3d-a507-3eb9ee9a8e68">
    <gml:name>Example Building LOD1</gml:name>
    ... (Attributes see example LOD1)
    <bldg:lod1Solid><gml:Solid>
      <gml:CompositeSurface gml:id="lod1Surface">
        <!-- Face Side 1 -->
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>7.0 7.0 0.0 17.0 0.0 17.0 7.0 0.0 17.0 7.0 4.0 7.0 7.0 0.0 7.0 7.0 4.0 7.0 7.0 0.0</gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        <!-- Face Side 2 -->
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>17.0 7.0 0.0 17.0 0.0 17.0 12.0 0.0 17.0 12.0 4.0 17.0 12.0 0.0 17.0 7.0 0.0 17.0 7.0 0.0</gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
        <!-- Face Side 3 -->
        <gml:surfaceMember>
          <gml:Polygon>
            <gml:exterior>
              <gml:LinearRing>
                <gml:posList>12.0 0.0 17.0 0.0 17.0 7.0 0.0 17.0 7.0 0.0 17.0 12.0 0.0 17.0 12.0 4.0 17.0 12.0 0.0 17.0 0.0 17.0 0.0</gml:posList>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
      </gml:Solid>
    </gml:CompositeSurface>
  </bldg:Building>
</cityObjectMember>
Listing 13: CityGML dataset illustrating the use of local coordinate reference systems. The dataset is visualised in Fig. 76 on page 272.
Annex H
(informative)

Example ADE for Noise Immission Simulation

This annex illustrates the usage of CityGML within an environmental simulation application. The definition of the corresponding Application Domain Extension (ADE) is included as an example.

The Environmental Noise Directive of the European Union 2002/49/EG obligates the EU member states to calculate every 5 years the noise levels at a height of 4m on buildings and to document the results in noise maps. The noise maps serve as information for the European Union and the citizens affected by noise (Fig. 94). These noise maps are generated on the basis of acoustic models and noise propagation calculations, not on the basis of measurements (Fig. 95). For the noise propagation calculations, a great number of thematic data and 3D geodata is necessary for each EU member state. Because of the large spatial extent of the noise calculation, the provision of statewide and ubiquitous 3D geodata on buildings, roads, railways and terrain for a multitude of users is necessary, in part with high requirements on resolution.

The calculation of noise levels from a road requires information about the traffic flow, the heavy vehicle percentage, the speed limits, the road surface types and the road gradient. Furthermore, the noise level depends on the distance between point of emission and reception (immission) as well as on reflection (e.g. on building facades) or shielding effects (e.g. noise barriers). The noise level is calculated separately for the day (06.00-18.00), the evening (18.00-22.00) and the night (22.00-06.00). As the noise level is calculated at a height of 4m and as the influence of vertical reflecting surfaces is considered (e.g. noise barriers and buildings), a multitude of geodata in the third dimension is necessary. In addition to all 3D geodata specific thematic data are required. For example the following data are necessary for the noise calculation of roads: Digital Terrain Model with 10m grid, 3D building models with their thematic attributes (e.g. reflection, inhabitants), 3D road data with their thematic attributes (e.g. traffic flow, heavy vehicle percentage, speed limit, type of road surface, road gradient, width of a road), 3D noise barriers and their thematic attributes (e.g. reflection).

![Fig. 94: Noise map generated from the 3D CityGML geodata and used for the reporting demanded of the EU Environmental Noise Directive (dark colours show higher noise immission) (source: State Agency for nature, environment and consumer protection NRW).]
Fig. 95: Modelling a noise emission source using the 3D CityGML geodata in a special noise calculation software as first step before generating the noise map in Fig. 94 (source: Surveying and Mapping Agency NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).

In the state of North Rhine-Westphalia, special conditions have to be considered: high population and transportation route density and therefore the highest amount of noise calculation areas and objects in Germany. The aim is to provide a sustainable, efficient and variable access to the required 3D geodata for the 5 years iteration period and the different noise calculation authorities.

In order to provide this considerable amount of statewide 3D geodata, the responsible partners, in particular the State Ministry of Environment, Nature Conservation, Agriculture and Consumer Protection of North Rhine-Westphalia, the State Agency for nature, environment and consumer protection of North Rhine-Westphalia and the Surveying and Mapping Agency of North Rhine-Westphalia, have decided to use the Spatial Data Infrastructure in North Rhine-Westphalia (GDI NRW) and to extend it with statewide Web Services for 2.5D and 3D geoinformation. Therefore, new OGC Web Services for building models, terrain, road and railway data were implemented (e.g. Web Feature Service for 3D block models in LOD1 and for 3D road and railway data, Web Coverage Service for DTM).

Fig. 96: 3D geodata in CityGML for the calculation of the noise map in Fig. 94: DTM in GeoTiff, 3D block model in CityGML, 3D road and railway data in CityGML, state road data for higher-level roads in CityGML (source: Surveying and Mapping Agency NRW, State Road Enterprise NRW, Stapelfeldt GmbH, Institute of Geodesy and Geoinformation Uni Bonn).
CityGML is used together with GeoTIFF as the only exchange formats between web services and noise calculation software (Fig. 96 - Fig. 98). For the special requirements of the noise directive, a CityGML noise application schema has been developed by the Institute of Geodesy and Geoinformation University of Bonn and the Special Interest Group SIG 3D of GDI NRW. It is based on the ADE mechanism (see chapter 6.12 and 10.13). This mechanism allows the supplementation of existing classes and objects in CityGML (e.g. buildings) by thematic attributes. The quantity as well as the type of these attributes is selectable. The CityGML schema can also be complemented by new classes. Hence, the noise application schema contains new objects (e.g. segmentation of roads according to noise requirements - *NoiseRoadSegment*, Fig. 99) as well as noise attributes attached to existing objects (e.g. reflection of buildings, Fig. 100). These additional noise attributes are derived from regulations issued by the Federal Government of Germany realising the obligations of the Environmental Noise Directive of the European Union (cf. BImSchV 2006, VBUS 2006, VBUSch 2006).

The interoperability techniques of this project demonstrate a remarkable innovation, as for the first time statewide 3D geodata are provided via common standards and web services (cf. Czerwinski et al. 2006b). Therefore, the Spatial Data Infrastructure for noise calculation in North Rhine-Westphalia provides an application example for the INSPIRE directive of the European Union 2007/2/EC (Infrastructure for Spatial Information in Europe) (cf. Czerwinski et al. 2007).
H.1 CityGML Noise ADE

In this section the data models for the CityGML Noise ADE are given as UML diagrams and XML schema. As the semantics of the specific attributes and object types result from the German regulations for noise immision computations, they are not explained in detail here (see BimSchV 2006, VBUS 2006, VBUSch 2006). The purpose of this section is to provide an example how CityGML can be extended using the ADE mechanism. The XML Schema definition of the CityGML Noise ADE together with example datasets can be obtained from http://schemas.opengis.net/citygml/examples/2.0/ade/noise-ade/.

Fig. 99: CityGML noise application schema – transportation model (light yellow=CityGML Transportation module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Transportation module. The prefix noise is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).
Fig. 100: CityGML noise application schema – excerpt of the building model (light yellow=CityGML Building module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Building module. The prefix noise is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).

Fig. 101: CityGML noise application schema – city furniture model (light yellow=CityGML CityFurniture module, light orange=CityGML Noise ADE). Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML CityFurniture module. The prefix noise is associated with the CityGML Noise ADE (source: Institute of Geodesy and Geoinformation Uni Bonn).
Header of the Noise ADE Schema definition file

```
<xs:complexType name="NoiseRoadSegmentPropertyType" substitutionGroup="tram:_GenericApplicationPropertyOfRoad" />
<xs:complexType name="NoiseCityFurnitureSegmentPropertyType" substitutionGroup="core:_CityObject" />
```

NoiseCityFurnitureSegmentTypeType, NoiseCityFurnitureSegment

```
<xs:complexType name="NoiseCityFurnitureSegmentPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="NoiseCityFurnitureSegment" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

NoiseRoadSegmentType, NoiseRoadSegment

```
<xs:complexType name="NoiseRoadSegmentPropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="NoiseRoadSegment"/>
  </xs:sequence>
</xs:complexType>
```
NoiseRailwaySegmentType, NoiseRailwaySegment

TrainType, TrainPropertyType
Application specific attributes for _AbstractBuilding

H.2 Example dataset

The following dataset illustrates a CityGML instance document which uses the application noise schema. It contains two CityObject features: a road object and a building object. The dataset references the XML Schema definition file of the CityGML Noise ADE which explicitly imports the XML Schema definitions of the CityGML modules extended by the Noise ADE (CityGML Core, Building, Transportation, and CityFurniture module). Thus, all classes defined by the employed CityGML modules can be used in the instance document. Furthermore, the application specific additions such as new object types (e.g. NoiseRoadSegment) and additional thematic attributes (e.g. the attributes defined for _AbstractBuilding) are available. These additional elements are distinguished from standard CityGML elements by the namespace prefix noise which refers to the noise schema definition.
Listing 14: Excerpt from a CityGML dataset implementing the illustrated CityGML noise application schema.
Annex I
(informative)

Example ADE for Ubiquitous Network Robots Services

This annex illustrates the usage of CityGML within Robotics services. The definition of the corresponding Application Domain Extension (ADE) is included as an example.

I.1 Overview of Ubiquitous Network Robots

In the field of robotics, there is a concept which originated in Japan called network robots, which aims to achieve advanced robot services by connecting multiple robots. The goal is to go beyond the capacity limits of a single robot. Sensors embedded in the environment and agents on the internet are also broadly regarded as robots, and the research and development of network robot technology that connects robots through a network has been promoted. Connecting multiple robots provides improved cognitive ability and interactive capability compared to a single robot, and it is thought that advanced robot services are achievable.

In Japan, the Network Robot Forum was established in 2003. Its goal is to smoothly and effectively advance the research and development of network robots and their standardization. There are approximately 150 members including machinery manufacturers, electronic manufacturers, telecommunications carriers, research institutes, academic experts, economic organizations, and local government. They engage in activities such as research, promotion, development, and standardization of network robots.

At a national level, the Ministry of Internal Affairs and Communications has started "Research and development of ubiquitous network robot technology (UNR) for the elderly and physically challenged" since 2009, and the research and development for ubiquitous network robots is fully in progress in Japan.

In the research and development projects for ubiquitous network robots, the extension of network robot services from a single point to multiple points is being considered. In order to expand the activities of network robots and provide various robot services, it is necessary to have a framework that controls the information about surrounding environments as spatial attributes and provides services in a way that robots can understand. In these research and development projects, the database that controls the spatial attributes for network robots is referred to as the spatial master database. On the assumption that robots move around inside and outside buildings, CityGML which is able to provide models for the representation of both outdoor and indoor features has been used in the spatial master database. However, there was a lack of essential information for mobile robots, such as storey structuring, flooring materials, and door attributes. Therefore, the CityGML data model was extended using an ADE mechanism. Fig. 102 shows an example 3D visualization of a spatial master database, which is generated from CityGML with Robotics ADE and is used in these research projects.

The coordinate system used for indoor positioning of mobile robots is not a global coordinate system as used outdoors, but a local coordinate system configured for each building. If a building has multiple floors, different coordinate systems might be applied for each floor. Furthermore, multiple coordinate systems might be applied for one floor. For interaction with humans, it is important to control multiple coordinate systems set up for each building and manage the semantic knowledge of individual objects by using a spatial master database.
Fig. 102: 3D spatial master database for a shopping mall. This was generated from CityGML with Robotics ADE and used for the ubiquitous network robots services research project in Japan. The height of each floor is set up when turning the original 2D CAD data into 3D data in the spatial master database. Thus, the heights of some equipment (for example, furniture) is not set up and is displayed in 2D. Also, the grid map (with obstacles information) generated by the grid map function of the spatial master database is attached to the ground on the floor surface. The grid map consists of grid cells of a given size, and is color-coded depending on whether obstacles exist or not. Robots use these grid maps to recognize whether they can pass through each grid (green=walls, yellow=pillars, blue=equipment, green floor=grid cells with obstacles such as pillars and furniture, red floor=grid cells without obstacles).

In the research and development project, a demonstration experiment was conducted at a commercial facility in 2010. Fig. 103 shows the experiment conducted at a shopping mall. Robots provided services such as talking with customers, showing customers around the store, and assisting with shopping. The shopping mall facility consists of multiple buildings, and services, in which robots move around inside and outside the buildings and show customers around the store, were also experimented with.

Spatial master database information was converted into a grid map (raster map) format, and was provided to the robots by a server. The grid map is a PNG image format, and four types of grid maps are supported: obstacles, flooring materials, floor slope, and floor bumps. The grid map is the flooring surface divided into grid cells of a given size and information for each grid cell is coded. The information of each grid cell is displayed by different color pixels in an image format of grid map. The information about flooring materials, floor slope, and floor bumps is required because such items can become impassable obstacles depending on the functions and constraints of the robot-moving mechanism. In the shopping mall used for the demonstration experiment, there are some high-angle slopes in the hallway between buildings. Two different types of robots were used in the experiment, one type was able to go over the slope and the other was not. The latter robot calls the former robot through a network and turned over the customer guidance, when it arrives at an impassable zone indicated in the grid maps.
I.2 Overview of the Spatial Master Database

This section gives an overview of the spatial master databases used for network robots. When the data model for a spatial master database is constructed, the compatibility with CityGML LOD4 is taken into consideration. Moreover, the concepts of flooring materials (materialType), Storey, and door attributes (doorOperationType), which are essential for mobile robots, are added. Section I.3 provides more details.

Fig. 104 shows the configuration of a spatial master database management system used in the demonstration experiment described in section I.1. This system imports information of a building, such as CAD data or CityGML data, and stores it in spatial master database. And when converting it into a grid map, this system searches the targeted 3D objects and projects them onto a 2D floor surface. The next step is to divide it into grid, designating floor number, floor area (Boundary Box) and grid size (width and height) that the floor surface is divided into. Finally, it generates PNG images whose pixel values are related to each grid information, such as flooring materials or whether some obstacles exist or not. After converting the 3D model to this grid map, this system provides it to the Area management gateway server.

The shopping mall where the demonstration experiment was conducted consists of two buildings (indoor) and a hallway (outdoor) that connects the buildings (shown in Fig. 105). The spatial master database mainly stores the indoor spatial data, and stores outdoor spatial data only for the hallway. Fig. 106 shows an example of a generated grid map for which the pixel values indicate flooring materials. The service providing the grid map is developed as an extended WMS interface. This WMS interface is based on the work which was developed in “Outdoor and Indoor 3D Routing Services Engineering Report” (cf. Sato 2009, OGC Doc. No. 09-067r2).
I.3 Overview of the CityGML ADE

This section describes the implementation of the spatial master database explained in section I.2 by using CityGML. Specifically, this section shows UML diagrams and XML schemas for the data model of the Robotics application schema that was implemented as two independent ADEs. The reason for preparing two ADEs is that there are two types of information which is needed for Ubiquitous Network Robots service. One ADE is for more general information which can be used not only for robots service but also for indoor service. The other ADE is for information which is focused on robots service. The first ADE is the CityGML Standard Opening ADE, which is a schema extending the _Opening, Door, and Window classes of CityGML. The second ADE is the UNR (Ubiquitous Network Robots) ADE, which is a schema adding classes such as Storey and attributes such as door type and floor materials that are essential for ubiquitous network robots.

The purpose of this section is to provide an example showing how CityGML can be extended using multiple ADEs. As the semantics of the specific attributes and object types, which are implemented as codelists, result from the Japanese national research project for Ubiquitous Network Robots services, they are not explained in detail here. This section explains the Standard Opening ADE and UNR ADE. ADE details are explained in the CityGML Wiki (see http://www.citygmlwiki.org/index.php/CityGML-ADEs). The XML Schema definition of both ADEs together with example datasets and code lists can be additionally obtained from http://schemas.opengis.net/citygml/examples/2.0/ade/robotics-ade/.
The storey number is given as "storeyNo_*" in the gml:name attribute of CityObjectGroup.

Door and Window are extended with attributes indicating how to open doors and windows.
Header of the Standard Openings ADE Schema definition file

```xml
<xsd:schema xmlns="http://unr.crl.hitachi.co.jp/ade/standard_opening"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:gml="http://www.opengis.net/gml"
    targetNamespace="http://unr.crl.hitachi.co.jp/ade/standard_opening" elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xsd:import namespace="http://www.opengis.net/gml"
        schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
    <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
        schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
    ...
</xsd:schema>
```

Application specific attributes for Door

```xml
<xsd:element name="doorOperationType" type="DoorOperationTypeType"
    substitutionGroup="bldg:_GenericApplicationPropertyOfDoor"/>
<xsd:simpleType name="DoorOperationTypeType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="swinging"/>
        <xsd:enumeration value="double_acting"/>
        <xsd:enumeration value="sliding"/>
        <xsd:enumeration value="folding"/>
        <xsd:enumeration value="revolving"/>
        <xsd:enumeration value="folding"/>
        <xsd:enumeration value="rollingup"/>
        <xsd:enumeration value="userdefined"/>
        <xsd:enumeration value="notdefined"/>
    </xsd:restriction>
</xsd:simpleType>
```

Application specific attributes for Window

```xml
<xsd:element name="windowOperationType" type="WindowOperationTypeType"
    substitutionGroup="bldg:_GenericApplicationPropertyOfDoor"/>
<xsd:simpleType name="WindowOperationTypeType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="sidehungrighthand"/>
        <xsd:enumeration value="sidehunglefthand"/>
        <xsd:enumeration value="tiltandturnrighthand"/>
        <xsd:enumeration value="tiltandturnlefthand"/>
        <xsd:enumeration value="tophung"/>
        <xsd:enumeration value="bottomhung"/>
        <xsd:enumeration value="pivothorizontal"/>
        <xsd:enumeration value="pivotvertical"/>
        <xsd:enumeration value="slidinghorizontal"/>
        <xsd:enumeration value="slidingvertical"/>
        <xsd:enumeration value="removablecasement"/>
        <xsd:enumeration value="fixedcasement"/>
        <xsd:enumeration value="otheroperation"/>
        <xsd:enumeration value="notdefined"/>
    </xsd:restriction>
</xsd:simpleType>
```

Header of the UNR ADE Schema definition file

```xml
<xsd:schema xmlns="http://unr.crl.hitachi.co.jp/ade/unr"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:core="http://www.opengis.net/citygml/2.0"
    xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
    targetNamespace="http://unr.crl.hitachi.co.jp/ade/unr" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xsd:import namespace="http://www.opengis.net/gml"
        schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
    <xsd:import namespace="http://www.opengis.net/citygml/building/2.0"
        schemaLocation="http://schemas.opengis.net/citygml/building/2.0/building.xsd"/>
    ...
</xsd:schema>
```
I.4 Example Dataset

The following dataset illustrates a CityGML instance document that uses the Robotics application schema. It contains three _CityObject features: a CityObjectGroup object with a Storey object property, a FloorSurface object with a material type attribute, and a Door object with a doorOperationType attribute. The dataset references the XML schema definition files of the Robotics ADE, which explicitly imports the XML schema definitions of the CityGML modules extended by the CityGML Standard Opening ADE (CityGML Core and Building module) and the UNR ADE (CityGML Core, Building, and CityObjectGroup module). Thus, all classes defined by the employed CityGML modules can be used in the instance document. Furthermore, the application specific additions such as new object types (e.g. Storey) and additional thematic attributes (e.g. the attributes defined for
Door) are available. These additional elements are distinguished from standard CityGML elements by the namespace prefix stdOp and unr which refer to the Robotics schema definition.

In order to use multiple ADE files, it is necessary to follow the XML schema rules and explicitly specify in the xsi:schemaLocation tag that multiple ADE files (of the XML schema) are to be read. In the following dataset, the two pairs of schema location, Standard Opening ADE and UNR ADE, are coded in xsi:schemaLocation. When software such as a CityGML parser is implemented, multiple XML schema files must be read. Also, when multiple ADE files are used, problems such as circular references might occur and should be avoided carefully.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<CityModel
xmlns='http://www.opengis.net/citygml/2.0'
xmlns:xlink='http://www.w3.org/1999/xlink'
xmlns:stdOp='http://www.opengis.net/gml'
xmlns:bldg='http://www.opengis.net/citygml/building/2.0'
xmlns:grp='http://www.opengis.net/citygml/cityobjectgroup/2.0'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:unr='http://unr.crl.hitachi.co.jp/ade/unr'
xmlns:unr='http://unr.crl.hitachi.co.jp/ade/unr'
<cityObjectMember>
  <grp:CityObjectGroup
gml:id='Storey_0'>
    <gml:name>Sample Storey 0</gml:name>
    <gml:name>storeyNo_0</gml:name>
    <grp:class>building separation</grp:class>
    <grp:function>lod4Storey</grp:function>
    <grp:groupMember xlink:href='#FloorSurface_1'/>
    <grp:groupMember xlink:href='#Door_1'/>
    <unr:storeyProperty>
      <unr:Storey>
        <unr:heightAboveGround uom='#m'>0.0</unr:heightAboveGround>
        <unr:heightToCeiling uom='#m'>5.0</unr:heightToCeiling>
      </unr:Storey>
    </unr:storeyProperty>
  </grp:CityObjectGroup>
</cityObjectMember>
<cityObjectMember>
  <bldg:Building>
    ...
    <bldg:boundedBy>
      <bldg:FloorSurface
gml:id='FloorSurface_1'>
        <gml:name>Sample FloorSurface 1</gml:name>
        <bldg:lod4MultiSurface>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
srsDimension='3'>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</gml:posList
</gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
srsDimension='3'>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</gml:posList
</gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
srsDimension='3'>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
</gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
srsDimension='3'>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
</gml:exterior>
            </gml:Polygon>
          </gml:surfaceMember>
          <gml:MultiSurface>
            <gml:surfaceMember>
              <gml:Polygon>
                <gml:exterior>
                  <gml:LinearRing>
                    <gml:posList
srsDimension='3'>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
</gml:exterior>
```
Listing 15: Excerpt from a CityGML dataset implementing the illustrated CityGML Robotics application schema. Refer to code lists included in the schema for information about attribute values such as 2491 of `unr:openingMaterialType`. 
## Annex K
(informative)

### Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Editor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-02-01</td>
<td>0.1.0</td>
<td>Czerwinski, Kolbe, Gröger</td>
<td>Initialisation of the document.</td>
</tr>
<tr>
<td>2006-02-22</td>
<td>0.1.0</td>
<td>Gröger, Gruber</td>
<td>Additions to UML diagrams.</td>
</tr>
<tr>
<td>2006-04-26</td>
<td>0.2.0</td>
<td>Kolbe, Gröger, Czerwinski</td>
<td>Release of CityGML draft standard to EuroSDR.</td>
</tr>
<tr>
<td>2006-03-06</td>
<td>0.3.0</td>
<td>Kolbe, Gröger, Czerwinski</td>
<td>Changes on property names, some attributes added. Release of CityGML standard document to OGC.</td>
</tr>
<tr>
<td>2007-05-30</td>
<td>0.4.0</td>
<td>Kolbe, Gröger, Nagel, Lorenz, Benner, Czerwinski, Gruber, Schlüter, Bildstein, Drees, Löwner</td>
<td>Introduction of a new appearance model. Introduction of Application Domain Extensions (ADE). Minor changes to the building model. Minor changes to city object groups. The concept of TerrainIntersectionCurves (TIC) added to CityFurniture. Adaptation of code lists. Release of CityGML standard document to OGC.</td>
</tr>
<tr>
<td>2008-02-04</td>
<td>1.0.0</td>
<td>Kolbe, Gröger, Nagel, Stadler, Lorenz</td>
<td>Introduction of modularisation of the CityGML data model. Minor changes to the appearance model. Minor changes to city object groups and transportation objects. Encoding of external code lists changed to GML 3.1.1 Simple Dictionary Profile. Revision of UML diagrams. Release of draft CityGML standard document to CityGML 1.0 SWG.</td>
</tr>
<tr>
<td>2008-05-19</td>
<td>1.0.0</td>
<td>Kolbe, Gröger, Nagel, Stadler, Lorenz</td>
<td>Incorporation of changes and editorial corrections based on comments and recommendations during the OGC RFC public comment phase and corresponding actions agreed on by the CityGML 1.0 SWG. Final release of draft CityGML standard document to CityGML 1.0 SWG.</td>
</tr>
<tr>
<td>2008-08-18</td>
<td>1.0.0</td>
<td>Carl Reed</td>
<td>Prepare for posting as OGC™ standard and make final edits.</td>
</tr>
<tr>
<td>2011-07-19</td>
<td>1.1.0</td>
<td>Nagel, Häfele, Benner, Gröger, Gruber, Schlüter</td>
<td>First release of draft CityGML standard document to CityGML SWG.</td>
</tr>
<tr>
<td>2011-11-21</td>
<td>1.1.0</td>
<td>Nagel, Häfele, Gröger, Kolbe</td>
<td>Incorporation of changes and editorial corrections based on comments and recommendations during the OGC RFC public comment phase and corresponding actions agreed on by the CityGML SWG. Release of draft CityGML standard document to CityGML SWG.</td>
</tr>
<tr>
<td>2012-02-27</td>
<td>2.0.0</td>
<td>Nagel, Häfele, Gröger, Kolbe</td>
<td>Version changed from 1.1.0 to 2.0.0 in order to conform to OGC versioning policy as defined in 135r11. Reference number of this document changed to 12-019 to reflect the major version number change. Release of draft CityGML standard document to CityGML SWG.</td>
</tr>
<tr>
<td>2012-03-14</td>
<td>2.0.0</td>
<td>Carl Reed</td>
<td>Make final edits for publication as an OGC standard</td>
</tr>
</tbody>
</table>
Bibliography


BImSchV (2006): Federal Government of Germany (eds.): 34. Verordnung zur Durchführung des Bundesimmisionsenschutzgesetzes - Verordnung über die Lärmkartierung [34. BImSchV], in der Fassung vom 6.3.06, in Kraft seit 16.3.06, in BGBl. I 106, Nr. 12, S. 516 (in German only).


