

OGC CITY GEOGRAPHY MARKUP LANGUAGE (CITYGML) PART 1: CONCEPTUAL MODEL STANDARD

STANDARD
Conceptual model

APPROVED

Version: 3.0.0

Submission Date: 2021-03-02

Approval Date: 2021-06-04

Publication Date: 2021-09-13

Editor: Thomas H. Kolbe, Tatjana Kutzner, Carl Stephen Smyth, Claus Nagel, Carsten Roensdorf, Charles Heazel

Notice: 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.

Suggested additions, changes and comments on this document are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

Copyright notice

Copyright © 2022 Open Geospatial Consortium
To obtain additional rights of use, visit <http://www.ogc.org/legal/>

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

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

CONTENTS

I. ABSTRACT	xxix
II. KEYWORDS	xxix
III. SECURITY CONSIDERATIONS	xxx
IV. SUBMITTING ORGANIZATIONS	xxxi
V. SUBMITTERS	xxxi
VI. PARTICIPANTS IN DEVELOPMENT	xxxii
VII. INTRODUCTION	xxxiii
VIII. ACKNOWLEDGEMENTS	xxxiv
1. SCOPE	2
2. CONFORMANCE	6
2.1. Conceptual Models	6
2.2. Implementation Specifications	6
2.3. Conformance Classes	7
3. NORMATIVE REFERENCES	9
4. TERMS, DEFINITIONS AND ABBREVIATED TERMS	12
4.1. Terms and definitions	12
4.2. Abbreviated terms	15
5. CONVENTIONS	19
5.1. Identifiers	19
5.2. UML Notation	19
6. OVERVIEW OF CITYGML	25
6.1. Modularization	25
6.2. General Modeling Principles	27
6.3. Representation of Spatial Properties	29
6.4. CityGML Core Model: Space Concept, Levels of Detail, Special Spatial Types	33
6.5. Appearances	39
6.6. Modeling Dynamic Data	40
6.7. Extending CityGML	43

7. CITYGML UML MODEL	46
7.1. Structural overview of requirements classes	46
7.2. Core	47
7.3. Appearance	65
7.4. CityFurniture	69
7.5. CityObjectGroup	73
7.6. Dynamizer	77
7.7. Generics	83
7.8. LandUse	90
7.9. PointCloud	94
7.10. Relief	96
7.11. Transportation	99
7.12. Vegetation	110
7.13. Versioning	115
7.14. WaterBody	118
7.15. Construction	123
7.16. Bridge	132
7.17. Building	139
7.18. Tunnel	148
8. CITYGML DATA DICTIONARY	157
8.1. ISO Classes	157
8.2. Core	166
8.3. Appearance	199
8.4. CityFurniture	209
8.5. CityObjectGroup	212
8.6. Dynamizer	215
8.7. Generics	229
8.8. LandUse	241
8.9. PointCloud	244
8.10. Relief	246
8.11. Transportation	253
8.12. Vegetation	280
8.13. Versioning	286
8.14. WaterBody	291
8.15. Construction	296
8.16. Bridge	321
8.17. Building	332
8.18. Tunnel	350
9. APPLICATION DOMAIN EXTENSION (ADE)	364
9.1. General Rules for ADEs	364
9.2. Defining New ADE Model Elements	364
9.3. Augmenting CityGML Feature Types with Additional ADE Properties	365
9.4. Encoding of ADEs	367
9.5. Requirements and Recommendations	367

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE	371
A.1. Introduction	371
A.2. Conformance Class Core	371
A.3. Conformance Class Appearance	373
A.4. Conformance Class CityFurniture	374
A.5. Conformance Class CityObjectGroup	376
A.6. Conformance Class Dynamizer	378
A.7. Conformance Class Generics	379
A.8. Conformance Class LandUse	381
A.9. Conformance Class PointCloud	382
A.10. Conformance Class Relief	384
A.11. Conformance Class Transportation	385
A.12. Conformance Class Vegetation	387
A.13. Conformance Class Versioning	388
A.14. Conformance Class WaterBody	390
A.15. Conformance Class Construction	391
A.16. Conformance Class Bridge	393
A.17. Conformance Class Building	394
A.18. Conformance Class Tunnel	396
A.19. Conformance Class ADE	398
ANNEX B (INFORMATIVE) GLOSSARY	401
B.1.	401
ANNEX C (INFORMATIVE) REVISION HISTORY	416
BIBLIOGRAPHY	418

LIST OF TABLES

Table 1 – CityGML Core Sections	48
Table 2 – Core space classes and their allowed thematic surface boundaries	48
Table 3 – ISO Classes used in CityGML	51
Table 4 – City Model and City Object classes used in Core	53
Table 5 – Space Classes used in Core	56
Table 6 – Geometry Classes used in Core	58
Table 7 – Additional Classes used in Core	60
Table 8 – Data types defined in Core (ApplicationSchema)	61
Table 9 – Primitive data types defined in Core (ApplicationSchema)	63
Table 10 – Enumerated classes defined in Core (ApplicationSchema)	63
Table 11 – Union types defined in Core (ApplicationSchema)	64

Table 12 – Code list classes defined in Core (ApplicationSchema)	64
Table 13 – Classes defined in Appearance (ApplicationSchema)	68
Table 14 – Data types defined in Appearance (ApplicationSchema)	68
Table 15 – Primitive data types defined in Appearance (ApplicationSchema)	69
Table 16 – Enumerated classes defined in Appearance (ApplicationSchema)	69
Table 17 – CityFurniture space classes and their allowed thematic surface boundaries	71
Table 18 – Classes defined in CityFurniture (ApplicationSchema)	72
Table 19 – Data types defined in CityFurniture (ApplicationSchema)	73
Table 20 – Code list classes defined in CityFurniture (ApplicationSchema)	73
Table 21 – CityObjectGroup space classes and their allowed thematic surface boundaries	75
Table 22 – Classes defined in CityObjectGroup (ApplicationSchema)	76
Table 23 – Data types defined in CityObjectGroup (ApplicationSchema)	77
Table 24 – Code list classes defined in CityObjectGroup (ApplicationSchema)	77
Table 25 – Classes defined in Dynamizer (ApplicationSchema)	81
Table 26 – Data types defined in Dynamizer (ApplicationSchema)	82
Table 27 – Enumerated classes defined in Dynamizer (ApplicationSchema)	82
Table 28 – Code list classes defined in Dynamizer (ApplicationSchema)	82
Table 29 – Generics space classes and their allowed thematic surface boundaries	86
Table 30 – Classes defined in Generics (ApplicationSchema)	88
Table 31 – Data types defined in Generics (ApplicationSchema)	88
Table 32 – Code list classes defined in Generics (ApplicationSchema)	89
Table 33 – Classes defined in LandUse (ApplicationSchema)	93
Table 34 – Data types defined in LandUse (ApplicationSchema)	93
Table 35 – Code list classes defined in LandUse (ApplicationSchema)	93
Table 36 – Classes defined in PointCloud (ApplicationSchema)	96
Table 37 – Data types defined in PointCloud (ApplicationSchema)	96
Table 38 – Classes defined in Relief (ApplicationSchema)	98
Table 39 – Data types defined in Relief (ApplicationSchema)	99
Table 40 – Transportation space classes and their allowed thematic surface boundaries	103
Table 41 – Classes defined in Transportation (ApplicationSchema)	105
Table 42 – Data types defined in Transportation (ApplicationSchema)	106
Table 43 – Enumerated classes defined in Transportation (ApplicationSchema)	107
Table 44 – Code list classes defined in Transportation (ApplicationSchema)	108
Table 45 – Vegetation space classes and their allowed thematic surface boundaries	112
Table 46 – Classes defined in Vegetation (ApplicationSchema)	114
Table 47 – Data types defined in Vegetation (ApplicationSchema)	114
Table 48 – Code list classes defined in Vegetation (ApplicationSchema)	114
Table 49 – Classes defined in Versioning (ApplicationSchema)	117
Table 50 – Data types defined in Versioning (ApplicationSchema)	118
Table 51 – Enumerated classes defined in Versioning (ApplicationSchema)	118
Table 52 – WaterBody space classes and their allowed thematic surface boundaries	120

Table 53 – Classes defined in WaterBody (ApplicationSchema)	121
Table 54 – Data types defined in WaterBody (ApplicationSchema)	122
Table 55 – Code list classes defined in WaterBody (ApplicationSchema)	122
Table 56 – Construction space classes and their allowed thematic surface boundaries	126
Table 57 – Classes defined in Construction (ApplicationSchema)	128
Table 58 – Data types defined in Construction (ApplicationSchema)	129
Table 59 – Enumerated classes defined in Construction (ApplicationSchema)	131
Table 60 – Code list classes defined in Construction (ApplicationSchema)	131
Table 61 – Bridge space classes and their allowed thematic surface boundaries	134
Table 62 – Classes defined in Bridge (ApplicationSchema)	137
Table 63 – Data types defined in Bridge (ApplicationSchema)	138
Table 64 – Code list classes defined in Bridge (ApplicationSchema)	138
Table 65 – Building space classes and their allowed thematic surface boundaries	142
Table 66 – Classes defined in Building (ApplicationSchema)	145
Table 67 – Data types defined in Building (ApplicationSchema)	146
Table 68 – Code list classes defined in Building (ApplicationSchema)	146
Table 69 – Tunnel space classes and their allowed thematic surface boundaries	150
Table 70 – Classes defined in Tunnel (ApplicationSchema)	153
Table 71 – Data types defined in Tunnel (ApplicationSchema)	154
Table 72 – Code list classes defined in Tunnel (ApplicationSchema)	154
Table 73 – Metadata of AnyFeature (class)	158
Table 74 – Associations of AnyFeature (class)	158
Table 75 – Metadata of CV_DiscreteGridPointCoverage (class)	158
Table 76 – Associations of CV_DiscreteGridPointCoverage (class)	158
Table 77 – Metadata of DirectPosition (class)	159
Table 78 – Associations of DirectPosition (class)	159
Table 79 – Attributes of DirectPosition (class)	159
Table 80 – Metadata of GM_Object (class)	160
Table 81 – Associations of GM_Object (class)	160
Table 82 – Metadata of GM_MultiCurve (class)	161
Table 83 – Attributes of GM_MultiCurve (class)	161
Table 84 – Metadata of GM_MultiPoint (class)	161
Table 85 – Attributes of GM_MultiPoint (class)	161
Table 86 – Metadata of GM_MultiSurface (class)	161
Table 87 – Attributes of GM_MultiSurface (class)	162
Table 88 – Metadata of GM_Point (class)	162
Table 89 – Associations of GM_Point (class)	162
Table 90 – Attributes of GM_Point (class)	162
Table 91 – Metadata of GM_Solid (class)	163
Table 92 – Associations of GM_Solid (class)	163
Table 93 – Metadata of GM_Surface (class)	163

Table 94 – Associations of GM_Surface (class)	164
Table 95 – Metadata of GM_Tin (class)	164
Table 96 – Attributes of GM_Tin (class)	164
Table 97 – Metadata of GM_TriangulatedSurface (class)	165
Table 98 – Metadata of SC_CRS (class)	165
Table 99 – Associations of SC_CRS (class)	165
Table 100 – Attributes of SC_CRS (class)	165
Table 101 – Metadata of TM_Position (class)	166
Table 102 – Attributes of TM_Position (class)	166
Table 103 – Metadata of Core (ApplicationSchema)	166
Table 104 – Metadata of AbstractAppearance (FeatureType)	167
Table 105 – Attributes of AbstractAppearance (FeatureType)	167
Table 106 – Metadata of AbstractCityObject (FeatureType)	167
Table 107 – Associations of AbstractCityObject (FeatureType)	167
Table 108 – Attributes of AbstractCityObject (FeatureType)	168
Table 109 – Metadata of AbstractDynamizer (FeatureType)	168
Table 110 – Attributes of AbstractDynamizer (FeatureType)	168
Table 111 – Metadata of AbstractFeature (FeatureType)	169
Table 112 – Attributes of AbstractFeature (FeatureType)	169
Table 113 – Metadata of AbstractFeatureWithLifespan (FeatureType)	169
Table 114 – Attributes of AbstractFeatureWithLifespan (FeatureType)	170
Table 115 – Metadata of AbstractLogicalSpace (FeatureType)	170
Table 116 – Attributes of AbstractLogicalSpace (FeatureType)	170
Table 117 – Metadata of AbstractOccupiedSpace (FeatureType)	171
Table 118 – Associations of AbstractOccupiedSpace (FeatureType)	171
Table 119 – Attributes of AbstractOccupiedSpace (FeatureType)	171
Table 120 – Metadata of AbstractPhysicalSpace (FeatureType)	171
Table 121 – Associations of AbstractPhysicalSpace (FeatureType)	172
Table 122 – Attributes of AbstractPhysicalSpace (FeatureType)	172
Table 123 – Metadata of AbstractPointCloud (FeatureType)	172
Table 124 – Attributes of AbstractPointCloud (FeatureType)	172
Table 125 – Metadata of AbstractSpace (FeatureType)	173
Table 126 – Associations of AbstractSpace (FeatureType)	173
Table 127 – Attributes of AbstractSpace (FeatureType)	174
Table 128 – Metadata of AbstractSpaceBoundary (FeatureType)	174
Table 129 – Attributes of AbstractSpaceBoundary (FeatureType)	174
Table 130 – Metadata of AbstractThematicSurface (FeatureType)	174
Table 131 – Associations of AbstractThematicSurface (FeatureType)	175
Table 132 – Attributes of AbstractThematicSurface (FeatureType)	175
Table 133 – Metadata of AbstractUnoccupiedSpace (FeatureType)	175
Table 134 – Attributes of AbstractUnoccupiedSpace (FeatureType)	176

Table 135 – Metadata of AbstractVersion (FeatureType)	176
Table 136 – Attributes of AbstractVersion (FeatureType)	176
Table 137 – Metadata of AbstractVersionTransition (FeatureType)	176
Table 138 – Attributes of AbstractVersionTransition (FeatureType)	177
Table 139 – Metadata of Address (FeatureType)	177
Table 140 – Associations of Address (FeatureType)	177
Table 141 – Attributes of Address (FeatureType)	177
Table 142 – Metadata of CityModel (FeatureType)	178
Table 143 – Associations of CityModel (FeatureType)	178
Table 144 – Attributes of CityModel (FeatureType)	178
Table 145 – Metadata of CityObjectRelation (ObjectType)	178
Table 146 – Associations of CityObjectRelation (ObjectType)	179
Table 147 – Attributes of CityObjectRelation (ObjectType)	179
Table 148 – Metadata of ClosureSurface (FeatureType)	179
Table 149 – Attributes of ClosureSurface (FeatureType)	179
Table 150 – Metadata of ImplicitGeometry (ObjectType)	179
Table 151 – Associations of ImplicitGeometry (ObjectType)	180
Table 152 – Attributes of ImplicitGeometry (ObjectType)	180
Table 153 – Metadata of Code (BasicType)	180
Table 154 – Attributes of Code (BasicType)	181
Table 155 – Metadata of DoubleBetween0and1 (BasicType)	181
Table 156 – Metadata of DoubleBetween0and1List (BasicType)	181
Table 157 – Attributes of DoubleBetween0and1List (BasicType)	181
Table 158 – Metadata of DoubleList (BasicType)	182
Table 159 – Attributes of DoubleList (BasicType)	182
Table 160 – Metadata of DoubleOrNilReasonList (BasicType)	182
Table 161 – Attributes of DoubleOrNilReasonList (BasicType)	182
Table 162 – Metadata of ID (BasicType)	183
Table 163 – Metadata of IntegerBetween0and3 (BasicType)	183
Table 164 – Metadata of MeasureOrNilReasonList (BasicType)	183
Table 165 – Attributes of MeasureOrNilReasonList (BasicType)	184
Table 166 – Metadata of TransformationMatrix2×2 (BasicType)	184
Table 167 – Metadata of TransformationMatrix3×4 (BasicType)	184
Table 168 – Metadata of TransformationMatrix4×4 (BasicType)	184
Table 169 – Metadata of CityModelMember (Union)	185
Table 170 – Attributes of CityModelMember (Union)	185
Table 171 – Metadata of DoubleOrNilReason (Union)	186
Table 172 – Attributes of DoubleOrNilReason (Union)	186
Table 173 – Metadata of NilReason (Union)	186
Table 174 – Attributes of NilReason (Union)	186
Table 175 – Metadata of IntervalValue (CodeList)	187

Table 176 – Metadata of MimeTypeValue (CodeList)	187
Table 177 – Metadata of NilReasonEnumeration (CodeList)	187
Table 178 – Metadata of OccupantTypeValue (CodeList)	188
Table 179 – Metadata of OtherRelationTypeValue (CodeList)	188
Table 180 – Metadata of QualifiedAreaTypeValue (CodeList)	188
Table 181 – Metadata of QualifiedVolumeTypeValue (CodeList)	188
Table 182 – Metadata of RelationTypeValue (CodeList)	189
Table 183 – Metadata of TemporalRelationTypeValue (CodeList)	189
Table 184 – Metadata of TopologicalRelationTypeValue (CodeList)	189
Table 185 – Metadata of AbstractGenericAttribute (DataType)	190
Table 186 – Metadata of ADEOfAbstractAppearance (DataType)	190
Table 187 – Metadata of ADEOfAbstractCityObject (DataType)	190
Table 188 – Metadata of ADEOfAbstractDynamizer (DataType)	190
Table 189 – Metadata of ADEOfAbstractFeature (DataType)	191
Table 190 – Metadata of ADEOfAbstractFeatureWithLifespan (DataType)	191
Table 191 – Metadata of ADEOfAbstractLogicalSpace (DataType)	191
Table 192 – Metadata of ADEOfAbstractOccupiedSpace (DataType)	191
Table 193 – Metadata of ADEOfAbstractPhysicalSpace (DataType)	192
Table 194 – Metadata of ADEOfAbstractPointCloud (DataType)	192
Table 195 – Metadata of ADEOfAbstractSpace (DataType)	192
Table 196 – Metadata of ADEOfAbstractSpaceBoundary (DataType)	193
Table 197 – Metadata of ADEOfAbstractThematicSurface (DataType)	193
Table 198 – Metadata of ADEOfAbstractUnoccupiedSpace (DataType)	193
Table 199 – Metadata of ADEOfAbstractVersion (DataType)	193
Table 200 – Metadata of ADEOfAbstractVersionTransition (DataType)	194
Table 201 – Metadata of ADEOfAddress (DataType)	194
Table 202 – Metadata of ADEOfCityModel (DataType)	194
Table 203 – Metadata of ADEOfClosureSurface (DataType)	194
Table 204 – Metadata of ExternalReference (DataType)	195
Table 205 – Attributes of ExternalReference (DataType)	195
Table 206 – Metadata of Occupancy (DataType)	195
Table 207 – Attributes of Occupancy (DataType)	196
Table 208 – Metadata of QualifiedArea (DataType)	196
Table 209 – Attributes of QualifiedArea (DataType)	196
Table 210 – Metadata of QualifiedVolume (DataType)	196
Table 211 – Attributes of QualifiedVolume (DataType)	197
Table 212 – Metadata of XALAddress (DataType)	197
Table 213 – Metadata of RelativeToTerrain (Enumeration)	197
Table 214 – Values of RelativeToTerrain (Enumeration)	197
Table 215 – Metadata of RelativeToWater (Enumeration)	198
Table 216 – Values of RelativeToWater (Enumeration)	198

Table 217 – Metadata of SpaceType (Enumeration)	199
Table 218 – Values of SpaceType (Enumeration)	199
Table 219 – Metadata of Appearance (ApplicationSchema)	199
Table 220 – Metadata of AbstractSurfaceData (FeatureType)	199
Table 221 – Attributes of AbstractSurfaceData (FeatureType)	200
Table 222 – Metadata of AbstractTexture (FeatureType)	200
Table 223 – Attributes of AbstractTexture (FeatureType)	200
Table 224 – Metadata of Appearance (FeatureType)	201
Table 225 – Associations of Appearance (FeatureType)	201
Table 226 – Attributes of Appearance (FeatureType)	201
Table 227 – Metadata of GeoreferencedTexture (FeatureType)	201
Table 228 – Associations of GeoreferencedTexture (FeatureType)	202
Table 229 – Attributes of GeoreferencedTexture (FeatureType)	202
Table 230 – Metadata of ParameterizedTexture (FeatureType)	202
Table 231 – Associations of ParameterizedTexture (FeatureType)	202
Table 232 – Attributes of ParameterizedTexture (FeatureType)	203
Table 233 – Metadata of TextureAssociation (ObjectType)	203
Table 234 – Attributes of TextureAssociation (ObjectType)	203
Table 235 – Metadata of X3DMaterial (FeatureType)	203
Table 236 – Attributes of X3DMaterial (FeatureType)	204
Table 237 – Metadata of Color (BasicType)	204
Table 238 – Metadata of ColorPlusOpacity (BasicType)	205
Table 239 – Metadata of AbstractTextureParameterization (DataType)	205
Table 240 – Metadata of ADEOfAbstractSurfaceData (DataType)	206
Table 241 – Metadata of ADEOfAbstractTexture (DataType)	206
Table 242 – Metadata of ADEOfAppearance (DataType)	206
Table 243 – Metadata of ADEOfGeoreferencedTexture (DataType)	206
Table 244 – Metadata of ADEOfParameterizedTexture (DataType)	207
Table 245 – Metadata of ADEOfX3DMaterial (DataType)	207
Table 246 – Metadata of TexCoordGen (DataType)	207
Table 247 – Associations of TexCoordGen (DataType)	207
Table 248 – Attributes of TexCoordGen (DataType)	208
Table 249 – Metadata of TexCoordList (DataType)	208
Table 250 – Attributes of TexCoordList (DataType)	208
Table 251 – Metadata of TextureType (Enumeration)	208
Table 252 – Values of TextureType (Enumeration)	209
Table 253 – Metadata of WrapMode (Enumeration)	209
Table 254 – Values of WrapMode (Enumeration)	209
Table 255 – Metadata of CityFurniture (ApplicationSchema)	210
Table 256 – Metadata of CityFurniture (TopLevelFeatureType)	210
Table 257 – Attributes of CityFurniture (TopLevelFeatureType)	210

Table 258 – Metadata of CityFurnitureClassValue (CodeList)	211
Table 259 – Metadata of CityFurnitureFunctionValue (CodeList)	211
Table 260 – Metadata of CityFurnitureUsageValue (CodeList)	211
Table 261 – Metadata of ADEOfCityFurniture (DataType)	212
Table 262 – Metadata of CityObjectGroup (ApplicationSchema)	212
Table 263 – Metadata of CityObjectGroup (TopLevelFeatureType)	213
Table 264 – Associations of CityObjectGroup (TopLevelFeatureType)	213
Table 265 – Attributes of CityObjectGroup (TopLevelFeatureType)	213
Table 266 – Metadata of Role (ObjectType)	213
Table 267 – Attributes of Role (ObjectType)	214
Table 268 – Metadata of CityObjectGroupClassValue (CodeList)	214
Table 269 – Metadata of CityObjectGroupFunctionValue (CodeList)	214
Table 270 – Metadata of CityObjectGroupUsageValue (CodeList)	215
Table 271 – Metadata of ADEOfCityObjectGroup (DataType)	215
Table 272 – Metadata of Dynamizer (ApplicationSchema)	215
Table 273 – Metadata of AbstractAtomicTimeseries (FeatureType)	216
Table 274 – Attributes of AbstractAtomicTimeseries (FeatureType)	216
Table 275 – Metadata of AbstractTimeseries (FeatureType)	217
Table 276 – Attributes of AbstractTimeseries (FeatureType)	217
Table 277 – Metadata of CompositeTimeseries (FeatureType)	217
Table 278 – Associations of CompositeTimeseries (FeatureType)	217
Table 279 – Attributes of CompositeTimeseries (FeatureType)	218
Table 280 – Metadata of Dynamizer (FeatureType)	218
Table 281 – Associations of Dynamizer (FeatureType)	218
Table 282 – Attributes of Dynamizer (FeatureType)	218
Table 283 – Metadata of GenericTimeseries (FeatureType)	219
Table 284 – Associations of GenericTimeseries (FeatureType)	219
Table 285 – Attributes of GenericTimeseries (FeatureType)	219
Table 286 – Metadata of StandardFileTimeseries (FeatureType)	220
Table 287 – Attributes of StandardFileTimeseries (FeatureType)	220
Table 288 – Metadata of TabulatedFileTimeseries (FeatureType)	220
Table 289 – Attributes of TabulatedFileTimeseries (FeatureType)	221
Table 290 – Metadata of AuthenticationTypeValue (CodeList)	222
Table 291 – Metadata of SensorConnectionTypeValue (CodeList)	222
Table 292 – Metadata of StandardFileTypeValue (CodeList)	223
Table 293 – Metadata of TabulatedFileTypeValue (CodeList)	223
Table 294 – Metadata of ADEOfAbstractAtomicTimeseries (DataType)	223
Table 295 – Metadata of ADEOfAbstractTimeseries (DataType)	224
Table 296 – Metadata of ADEOfCompositeTimeseries (DataType)	224
Table 297 – Metadata of ADEOfDynamizer (DataType)	224
Table 298 – Metadata of ADEOfGenericTimeseries (DataType)	224

Table 299 – Metadata of ADEOfStandardFileTimeseries (DataType)	225
Table 300 – Metadata of ADEOfTabulatedFileTimeseries (DataType)	225
Table 301 – Metadata of SensorConnection (DataType)	225
Table 302 – Associations of SensorConnection (DataType)	226
Table 303 – Attributes of SensorConnection (DataType)	226
Table 304 – Metadata of TimeseriesComponent (DataType)	227
Table 305 – Associations of TimeseriesComponent (DataType)	227
Table 306 – Attributes of TimeseriesComponent (DataType)	227
Table 307 – Metadata of TimeValuePair (DataType)	227
Table 308 – Attributes of TimeValuePair (DataType)	228
Table 309 – Metadata of TimeseriesTypeValue (Enumeration)	228
Table 310 – Values of TimeseriesTypeValue (Enumeration)	229
Table 311 – Metadata of Generics (ApplicationSchema)	229
Table 312 – Metadata of GenericLogicalSpace (TopLevelFeatureType)	230
Table 313 – Attributes of GenericLogicalSpace (TopLevelFeatureType)	230
Table 314 – Metadata of GenericOccupiedSpace (TopLevelFeatureType)	230
Table 315 – Attributes of GenericOccupiedSpace (TopLevelFeatureType)	230
Table 316 – Metadata of GenericThematicSurface (TopLevelFeatureType)	231
Table 317 – Attributes of GenericThematicSurface (TopLevelFeatureType)	231
Table 318 – Metadata of GenericUnoccupiedSpace (TopLevelFeatureType)	232
Table 319 – Attributes of GenericUnoccupiedSpace (TopLevelFeatureType)	232
Table 320 – Metadata of GenericLogicalSpaceClassValue (CodeList)	232
Table 321 – Metadata of GenericLogicalSpaceFunctionValue (CodeList)	233
Table 322 – Metadata of GenericLogicalSpaceUsageValue (CodeList)	233
Table 323 – Metadata of GenericOccupiedSpaceClassValue (CodeList)	233
Table 324 – Metadata of GenericOccupiedSpaceFunctionValue (CodeList)	234
Table 325 – Metadata of GenericOccupiedSpaceUsageValue (CodeList)	234
Table 326 – Metadata of GenericThematicSurfaceClassValue (CodeList)	234
Table 327 – Metadata of GenericThematicSurfaceFunctionValue (CodeList)	234
Table 328 – Metadata of GenericThematicSurfaceUsageValue (CodeList)	235
Table 329 – Metadata of GenericUnoccupiedSpaceClassValue (CodeList)	235
Table 330 – Metadata of GenericUnoccupiedSpaceFunctionValue (CodeList)	235
Table 331 – Metadata of GenericUnoccupiedSpaceUsageValue (CodeList)	235
Table 332 – Metadata of ADEOfGenericLogicalSpace (DataType)	236
Table 333 – Metadata of ADEOfGenericOccupiedSpace (DataType)	236
Table 334 – Metadata of ADEOfGenericThematicSurface (DataType)	236
Table 335 – Metadata of ADEOfGenericUnoccupiedSpace (DataType)	237
Table 336 – Metadata of CodeAttribute (DataType)	237
Table 337 – Attributes of CodeAttribute (DataType)	237
Table 338 – Metadata of DateAttribute (DataType)	237
Table 339 – Attributes of DateAttribute (DataType)	238

Table 340 – Metadata of DoubleAttribute (DataType)	238
Table 341 – Attributes of DoubleAttribute (DataType)	238
Table 342 – Metadata of GenericAttributeSet (DataType)	238
Table 343 – Associations of GenericAttributeSet (DataType)	239
Table 344 – Attributes of GenericAttributeSet (DataType)	239
Table 345 – Metadata of IntAttribute (DataType)	239
Table 346 – Attributes of IntAttribute (DataType)	239
Table 347 – Metadata of MeasureAttribute (DataType)	240
Table 348 – Attributes of MeasureAttribute (DataType)	240
Table 349 – Metadata of StringAttribute (DataType)	240
Table 350 – Attributes of StringAttribute (DataType)	240
Table 351 – Metadata of UriAttribute (DataType)	241
Table 352 – Attributes of UriAttribute (DataType)	241
Table 353 – Metadata of LandUse (ApplicationSchema)	241
Table 354 – Metadata of LandUse (TopLevelFeatureType)	242
Table 355 – Attributes of LandUse (TopLevelFeatureType)	242
Table 356 – Metadata of LandUseClassValue (CodeList)	243
Table 357 – Metadata of LandUseFunctionValue (CodeList)	243
Table 358 – Metadata of LandUseUsageValue (CodeList)	243
Table 359 – Metadata of ADEOfLandUse (DataType)	244
Table 360 – Metadata of PointCloud (ApplicationSchema)	244
Table 361 – Metadata of PointCloud (FeatureType)	244
Table 362 – Associations of PointCloud (FeatureType)	245
Table 363 – Attributes of PointCloud (FeatureType)	245
Table 364 – Metadata of ADEOfPointCloud (DataType)	246
Table 365 – Metadata of Relief (ApplicationSchema)	246
Table 366 – Metadata of AbstractReliefComponent (FeatureType)	247
Table 367 – Associations of AbstractReliefComponent (FeatureType)	247
Table 368 – Attributes of AbstractReliefComponent (FeatureType)	247
Table 369 – Metadata of BreaklineRelief (FeatureType)	247
Table 370 – Associations of BreaklineRelief (FeatureType)	248
Table 371 – Attributes of BreaklineRelief (FeatureType)	248
Table 372 – Metadata of MassPointRelief (FeatureType)	248
Table 373 – Associations of MassPointRelief (FeatureType)	248
Table 374 – Attributes of MassPointRelief (FeatureType)	248
Table 375 – Metadata of RasterRelief (FeatureType)	249
Table 376 – Associations of RasterRelief (FeatureType)	249
Table 377 – Attributes of RasterRelief (FeatureType)	249
Table 378 – Metadata of ReliefFeature (TopLevelFeatureType)	249
Table 379 – Associations of ReliefFeature (TopLevelFeatureType)	250
Table 380 – Attributes of ReliefFeature (TopLevelFeatureType)	250

Table 381 – Metadata of TINRelief (FeatureType)	250
Table 382 – Associations of TINRelief (FeatureType)	250
Table 383 – Attributes of TINRelief (FeatureType)	250
Table 384 – Metadata of ADEOfAbstractReliefComponent (DataType)	251
Table 385 – Metadata of ADEOfBreaklineRelief (DataType)	251
Table 386 – Metadata of ADEOfMassPointRelief (DataType)	252
Table 387 – Metadata of ADEOfRasterRelief (DataType)	252
Table 388 – Metadata of ADEOfReliefFeature (DataType)	252
Table 389 – Metadata of ADEOfTINRelief (DataType)	252
Table 390 – Metadata of Transportation (ApplicationSchema)	253
Table 391 – Metadata of AbstractTransportationSpace (FeatureType)	253
Table 392 – Associations of AbstractTransportationSpace (FeatureType)	254
Table 393 – Attributes of AbstractTransportationSpace (FeatureType)	254
Table 394 – Metadata of AuxiliaryTrafficArea (FeatureType)	254
Table 395 – Attributes of AuxiliaryTrafficArea (FeatureType)	254
Table 396 – Metadata of AuxiliaryTrafficSpace (FeatureType)	255
Table 397 – Associations of AuxiliaryTrafficSpace (FeatureType)	255
Table 398 – Attributes of AuxiliaryTrafficSpace (FeatureType)	255
Table 399 – Metadata of ClearanceSpace (FeatureType)	256
Table 400 – Attributes of ClearanceSpace (FeatureType)	256
Table 401 – Metadata of Hole (FeatureType)	256
Table 402 – Associations of Hole (FeatureType)	257
Table 403 – Attributes of Hole (FeatureType)	257
Table 404 – Metadata of HoleSurface (FeatureType)	257
Table 405 – Attributes of HoleSurface (FeatureType)	257
Table 406 – Metadata of Intersection (FeatureType)	258
Table 407 – Attributes of Intersection (FeatureType)	258
Table 408 – Metadata of Marking (FeatureType)	258
Table 409 – Attributes of Marking (FeatureType)	258
Table 410 – Metadata of Railway (TopLevelFeatureType)	259
Table 411 – Associations of Railway (TopLevelFeatureType)	259
Table 412 – Attributes of Railway (TopLevelFeatureType)	259
Table 413 – Metadata of Road (TopLevelFeatureType)	259
Table 414 – Associations of Road (TopLevelFeatureType)	260
Table 415 – Attributes of Road (TopLevelFeatureType)	260
Table 416 – Metadata of Section (FeatureType)	260
Table 417 – Attributes of Section (FeatureType)	261
Table 418 – Metadata of Square (TopLevelFeatureType)	261
Table 419 – Attributes of Square (TopLevelFeatureType)	261
Table 420 – Metadata of Track (TopLevelFeatureType)	261
Table 421 – Associations of Track (TopLevelFeatureType)	262

Table 422 – Attributes of Track (TopLevelFeatureType)	262
Table 423 – Metadata of TrafficArea (FeatureType)	262
Table 424 – Attributes of TrafficArea (FeatureType)	263
Table 425 – Metadata of TrafficSpace (FeatureType)	263
Table 426 – Associations of TrafficSpace (FeatureType)	263
Table 427 – Attributes of TrafficSpace (FeatureType)	264
Table 428 – Metadata of Waterway (TopLevelFeatureType)	264
Table 429 – Associations of Waterway (TopLevelFeatureType)	264
Table 430 – Attributes of Waterway (TopLevelFeatureType)	265
Table 431 – Metadata of AuxiliaryTrafficAreaClassValue (CodeList)	265
Table 432 – Metadata of AuxiliaryTrafficAreaFunctionValue (CodeList)	266
Table 433 – Metadata of AuxiliaryTrafficAreaUsageValue (CodeList)	266
Table 434 – Metadata of AuxiliaryTrafficSpaceClassValue (CodeList)	266
Table 435 – Metadata of AuxiliaryTrafficSpaceFunctionValue (CodeList)	266
Table 436 – Metadata of AuxiliaryTrafficSpaceUsageValue (CodeList)	267
Table 437 – Metadata of ClearanceSpaceClassValue (CodeList)	267
Table 438 – Metadata of HoleClassValue (CodeList)	267
Table 439 – Metadata of IntersectionClassValue (CodeList)	267
Table 440 – Metadata of MarkingClassValue (CodeList)	268
Table 441 – Metadata of RailwayClassValue (CodeList)	268
Table 442 – Metadata of RailwayFunctionValue (CodeList)	268
Table 443 – Metadata of RailwayUsageValue (CodeList)	269
Table 444 – Metadata of RoadClassValue (CodeList)	269
Table 445 – Metadata of RoadFunctionValue (CodeList)	269
Table 446 – Metadata of RoadUsageValue (CodeList)	269
Table 447 – Metadata of SectionClassValue (CodeList)	270
Table 448 – Metadata of SquareClassValue (CodeList)	270
Table 449 – Metadata of SquareFunctionValue (CodeList)	270
Table 450 – Metadata of SquareUsageValue (CodeList)	270
Table 451 – Metadata of SurfaceMaterialValue (CodeList)	271
Table 452 – Metadata of TrackClassValue (CodeList)	271
Table 453 – Metadata of TrackFunctionValue (CodeList)	271
Table 454 – Metadata of TrackUsageValue (CodeList)	272
Table 455 – Metadata of TrafficAreaClassValue (CodeList)	272
Table 456 – Metadata of TrafficAreaFunctionValue (CodeList)	272
Table 457 – Metadata of TrafficAreaUsageValue (CodeList)	272
Table 458 – Metadata of TrafficSpaceClassValue (CodeList)	273
Table 459 – Metadata of TrafficSpaceFunctionValue (CodeList)	273
Table 460 – Metadata of TrafficSpaceUsageValue (CodeList)	273
Table 461 – Metadata of WaterwayClassValue (CodeList)	273
Table 462 – Metadata of WaterwayFunctionValue (CodeList)	274

Table 463 – Metadata of WaterwayUsageValue (CodeList)	274
Table 464 – Metadata of ADEOfAbstractTransportationSpace (DataType)	274
Table 465 – Metadata of ADEOfAuxiliaryTrafficArea (DataType)	275
Table 466 – Metadata of ADEOfAuxiliaryTrafficSpace (DataType)	275
Table 467 – Metadata of ADEOfClearanceSpace (DataType)	275
Table 468 – Metadata of ADEOfHole (DataType)	275
Table 469 – Metadata of ADEOfHoleSurface (DataType)	276
Table 470 – Metadata of ADEOfIntersection (DataType)	276
Table 471 – Metadata of ADEOfMarking (DataType)	276
Table 472 – Metadata of ADEOfRailway (DataType)	277
Table 473 – Metadata of ADEOfRoad (DataType)	277
Table 474 – Metadata of ADEOfSection (DataType)	277
Table 475 – Metadata of ADEOfSquare (DataType)	277
Table 476 – Metadata of ADEOfTrack (DataType)	278
Table 477 – Metadata of ADEOfTrafficArea (DataType)	278
Table 478 – Metadata of ADEOfTrafficSpace (DataType)	278
Table 479 – Metadata of ADEOfWaterway (DataType)	278
Table 480 – Metadata of GranularityValue (Enumeration)	279
Table 481 – Values of GranularityValue (Enumeration)	279
Table 482 – Metadata of TrafficDirectionValue (Enumeration)	279
Table 483 – Values of TrafficDirectionValue (Enumeration)	279
Table 484 – Metadata of Vegetation (ApplicationSchema)	280
Table 485 – Metadata of AbstractVegetationObject (FeatureType)	280
Table 486 – Attributes of AbstractVegetationObject (FeatureType)	280
Table 487 – Metadata of PlantCover (TopLevelFeatureType)	281
Table 488 – Attributes of PlantCover (TopLevelFeatureType)	281
Table 489 – Metadata of SolitaryVegetationObject (TopLevelFeatureType)	281
Table 490 – Attributes of SolitaryVegetationObject (TopLevelFeatureType)	282
Table 491 – Metadata of PlantCoverClassValue (CodeList)	283
Table 492 – Metadata of PlantCoverFunctionValue (CodeList)	283
Table 493 – Metadata of PlantCoverUsageValue (CodeList)	283
Table 494 – Metadata of SolitaryVegetationObjectClassValue (CodeList)	284
Table 495 – Metadata of SolitaryVegetationObjectFunctionValue (CodeList)	284
Table 496 – Metadata of SolitaryVegetationObjectUsageValue (CodeList)	284
Table 497 – Metadata of SpeciesValue (CodeList)	284
Table 498 – Metadata of ADEOfAbstractVegetationObject (DataType)	285
Table 499 – Metadata of ADEOfPlantCover (DataType)	285
Table 500 – Metadata of ADEOfSolitaryVegetationObject (DataType)	285
Table 501 – Metadata of Versioning (ApplicationSchema)	286
Table 502 – Metadata of Version (FeatureType)	286
Table 503 – Associations of Version (FeatureType)	286

Table 504 – Attributes of Version (FeatureType)	286
Table 505 – Metadata of VersionTransition (FeatureType)	287
Table 506 – Associations of VersionTransition (FeatureType)	287
Table 507 – Attributes of VersionTransition (FeatureType)	287
Table 508 – Metadata of ADEOfVersion (DataType)	288
Table 509 – Metadata of ADEOfVersionTransition (DataType)	288
Table 510 – Metadata of Transaction (DataType)	289
Table 511 – Associations of Transaction (DataType)	289
Table 512 – Attributes of Transaction (DataType)	289
Table 513 – Metadata of TransactionTypeValue (Enumeration)	289
Table 514 – Values of TransactionTypeValue (Enumeration)	290
Table 515 – Metadata of TransitionTypeValue (Enumeration)	290
Table 516 – Values of TransitionTypeValue (Enumeration)	290
Table 517 – Metadata of WaterBody (ApplicationSchema)	291
Table 518 – Metadata of AbstractWaterBoundarySurface (FeatureType)	291
Table 519 – Attributes of AbstractWaterBoundarySurface (FeatureType)	291
Table 520 – Metadata of WaterBody (TopLevelFeatureType)	292
Table 521 – Associations of WaterBody (TopLevelFeatureType)	292
Table 522 – Attributes of WaterBody (TopLevelFeatureType)	292
Table 523 – Metadata of WaterGroundSurface (FeatureType)	292
Table 524 – Attributes of WaterGroundSurface (FeatureType)	293
Table 525 – Metadata of WaterSurface (FeatureType)	293
Table 526 – Attributes of WaterSurface (FeatureType)	293
Table 527 – Metadata of WaterBodyClassValue (CodeList)	294
Table 528 – Metadata of WaterBodyFunctionValue (CodeList)	294
Table 529 – Metadata of WaterBodyUsageValue (CodeList)	294
Table 530 – Metadata of WaterLevelValue (CodeList)	295
Table 531 – Metadata of ADEOfAbstractWaterBoundarySurface (DataType)	295
Table 532 – Metadata of ADEOfWaterBody (DataType)	295
Table 533 – Metadata of ADEOfWaterGroundSurface (DataType)	295
Table 534 – Metadata of ADEOfWaterSurface (DataType)	296
Table 535 – Metadata of Construction (ApplicationSchema)	296
Table 536 – Metadata of AbstractConstruction (FeatureType)	296
Table 537 – Associations of AbstractConstruction (FeatureType)	297
Table 538 – Attributes of AbstractConstruction (FeatureType)	297
Table 539 – Metadata of AbstractConstructionSurface (FeatureType)	298
Table 540 – Associations of AbstractConstructionSurface (FeatureType)	298
Table 541 – Attributes of AbstractConstructionSurface (FeatureType)	298
Table 542 – Metadata of AbstractConstructiveElement (FeatureType)	298
Table 543 – Associations of AbstractConstructiveElement (FeatureType)	298
Table 544 – Attributes of AbstractConstructiveElement (FeatureType)	299

Table 545 – Metadata of AbstractFillingElement (FeatureType)	299
Table 546 – Attributes of AbstractFillingElement (FeatureType)	299
Table 547 – Metadata of AbstractFillingSurface (FeatureType)	300
Table 548 – Attributes of AbstractFillingSurface (FeatureType)	300
Table 549 – Metadata of AbstractFurniture (FeatureType)	300
Table 550 – Attributes of AbstractFurniture (FeatureType)	300
Table 551 – Metadata of AbstractInstallation (FeatureType)	300
Table 552 – Associations of AbstractInstallation (FeatureType)	301
Table 553 – Attributes of AbstractInstallation (FeatureType)	301
Table 554 – Metadata of CeilingSurface (FeatureType)	301
Table 555 – Attributes of CeilingSurface (FeatureType)	301
Table 556 – Metadata of Door (FeatureType)	302
Table 557 – Associations of Door (FeatureType)	302
Table 558 – Attributes of Door (FeatureType)	302
Table 559 – Metadata of DoorSurface (FeatureType)	302
Table 560 – Associations of DoorSurface (FeatureType)	303
Table 561 – Attributes of DoorSurface (FeatureType)	303
Table 562 – Metadata of FloorSurface (FeatureType)	303
Table 563 – Attributes of FloorSurface (FeatureType)	303
Table 564 – Metadata of GroundSurface (FeatureType)	304
Table 565 – Attributes of GroundSurface (FeatureType)	304
Table 566 – Metadata of InteriorWallSurface (FeatureType)	304
Table 567 – Attributes of InteriorWallSurface (FeatureType)	304
Table 568 – Metadata of OtherConstruction (TopLevelFeatureType)	304
Table 569 – Attributes of OtherConstruction (TopLevelFeatureType)	305
Table 570 – Metadata of OuterCeilingSurface (FeatureType)	305
Table 571 – Attributes of OuterCeilingSurface (FeatureType)	305
Table 572 – Metadata of OuterFloorSurface (FeatureType)	306
Table 573 – Attributes of OuterFloorSurface (FeatureType)	306
Table 574 – Metadata of RoofSurface (FeatureType)	306
Table 575 – Attributes of RoofSurface (FeatureType)	306
Table 576 – Metadata of WallSurface (FeatureType)	306
Table 577 – Attributes of WallSurface (FeatureType)	307
Table 578 – Metadata of Window (FeatureType)	307
Table 579 – Associations of Window (FeatureType)	307
Table 580 – Attributes of Window (FeatureType)	307
Table 581 – Metadata of WindowSurface (FeatureType)	308
Table 582 – Attributes of WindowSurface (FeatureType)	308
Table 583 – Metadata of DoorClassName (CodeList)	309
Table 584 – Metadata of DoorFunctionName (CodeList)	309
Table 585 – Metadata of DoorUsageName (CodeList)	309

Table 586 – Metadata of ElevationReferenceValue (CodeList)	309
Table 587 – Metadata of EventValue (CodeList)	310
Table 588 – Metadata of OtherConstructionClassValue (CodeList)	310
Table 589 – Metadata of OtherConstructionFunctionValue (CodeList)	310
Table 590 – Metadata of OtherConstructionUsageValue (CodeList)	310
Table 591 – Metadata of WindowClassValue (CodeList)	311
Table 592 – Metadata of WindowFunctionValue (CodeList)	311
Table 593 – Metadata of WindowUsageValue (CodeList)	311
Table 594 – Metadata of ADEOfAbstractConstruction (DataType)	312
Table 595 – Metadata of ADEOfAbstractConstructionSurface (DataType)	312
Table 596 – Metadata of ADEOfAbstractConstructiveElement (DataType)	312
Table 597 – Metadata of ADEOfAbstractFillingElement (DataType)	313
Table 598 – Metadata of ADEOfAbstractFillingSurface (DataType)	313
Table 599 – Metadata of ADEOfAbstractFurniture (DataType)	313
Table 600 – Metadata of ADEOfAbstractInstallation (DataType)	313
Table 601 – Metadata of ADEOfCeilingSurface (DataType)	314
Table 602 – Metadata of ADEOfDoor (DataType)	314
Table 603 – Metadata of ADEOfDoorSurface (DataType)	314
Table 604 – Metadata of ADEOfFloorSurface (DataType)	314
Table 605 – Metadata of ADEOfGroundSurface (DataType)	315
Table 606 – Metadata of ADEOfInteriorWallSurface (DataType)	315
Table 607 – Metadata of ADEOfOtherConstruction (DataType)	315
Table 608 – Metadata of ADEOfOuterCeilingSurface (DataType)	316
Table 609 – Metadata of ADEOfOuterFloorSurface (DataType)	316
Table 610 – Metadata of ADEOfRoofSurface (DataType)	316
Table 611 – Metadata of ADEOfWallSurface (DataType)	316
Table 612 – Metadata of ADEOfWindow (DataType)	317
Table 613 – Metadata of ADEOfWindowSurface (DataType)	317
Table 614 – Metadata of ConstructionEvent (DataType)	317
Table 615 – Attributes of ConstructionEvent (DataType)	317
Table 616 – Metadata of Elevation (DataType)	318
Table 617 – Attributes of Elevation (DataType)	318
Table 618 – Metadata of Height (DataType)	318
Table 619 – Attributes of Height (DataType)	319
Table 620 – Metadata of ConditionOfConstructionValue (Enumeration)	319
Table 621 – Values of ConditionOfConstructionValue (Enumeration)	319
Table 622 – Metadata of HeightStatusValue (Enumeration)	320
Table 623 – Values of HeightStatusValue (Enumeration)	320
Table 624 – Metadata of RelationToConstruction (Enumeration)	320
Table 625 – Values of RelationToConstruction (Enumeration)	320
Table 626 – Metadata of Bridge (ApplicationSchema)	321

Table 627 – Metadata of AbstractBridge (FeatureType)	321
Table 628 – Associations of AbstractBridge (FeatureType)	321
Table 629 – Attributes of AbstractBridge (FeatureType)	322
Table 630 – Metadata of Bridge (TopLevelFeatureType)	322
Table 631 – Associations of Bridge (TopLevelFeatureType)	322
Table 632 – Attributes of Bridge (TopLevelFeatureType)	322
Table 633 – Metadata of BridgeConstructiveElement (FeatureType)	323
Table 634 – Attributes of BridgeConstructiveElement (FeatureType)	323
Table 635 – Metadata of BridgeFurniture (FeatureType)	323
Table 636 – Attributes of BridgeFurniture (FeatureType)	324
Table 637 – Metadata of BridgeInstallation (FeatureType)	324
Table 638 – Attributes of BridgeInstallation (FeatureType)	324
Table 639 – Metadata of BridgePart (FeatureType)	325
Table 640 – Attributes of BridgePart (FeatureType)	325
Table 641 – Metadata of BridgeRoom (FeatureType)	325
Table 642 – Associations of BridgeRoom (FeatureType)	325
Table 643 – Attributes of BridgeRoom (FeatureType)	326
Table 644 – Metadata of BridgeClassValue (CodeList)	326
Table 645 – Metadata of BridgeConstructiveElementClassValue (CodeList)	327
Table 646 – Metadata of BridgeConstructiveElementFunctionValue (CodeList)	327
Table 647 – Metadata of BridgeConstructiveElementUsageValue (CodeList)	327
Table 648 – Metadata of BridgeFunctionValue (CodeList)	327
Table 649 – Metadata of BridgeFurnitureClassValue (CodeList)	328
Table 650 – Metadata of BridgeFurnitureFunctionValue (CodeList)	328
Table 651 – Metadata of BridgeFurnitureUsageValue (CodeList)	328
Table 652 – Metadata of BridgeInstallationClassValue (CodeList)	328
Table 653 – Metadata of BridgeInstallationFunctionValue (CodeList)	329
Table 654 – Metadata of BridgeInstallationUsageValue (CodeList)	329
Table 655 – Metadata of BridgeRoomClassValue (CodeList)	329
Table 656 – Metadata of BridgeRoomFunctionValue (CodeList)	330
Table 657 – Metadata of BridgeRoomUsageValue (CodeList)	330
Table 658 – Metadata of BridgeUsageValue (CodeList)	330
Table 659 – Metadata of ADEOfAbstractBridge (DataType)	330
Table 660 – Metadata of ADEOfBridge (DataType)	331
Table 661 – Metadata of ADEOfBridgeConstructiveElement (DataType)	331
Table 662 – Metadata of ADEOfBridgeFurniture (DataType)	331
Table 663 – Metadata of ADEOfBridgeInstallation (DataType)	332
Table 664 – Metadata of ADEOfBridgePart (DataType)	332
Table 665 – Metadata of ADEOfBridgeRoom (DataType)	332
Table 666 – Metadata of Building (ApplicationSchema)	333
Table 667 – Metadata of AbstractBuilding (FeatureType)	333

Table 668 – Associations of AbstractBuilding (FeatureType)	333
Table 669 – Attributes of AbstractBuilding (FeatureType)	334
Table 670 – Metadata of AbstractBuildingSubdivision (FeatureType)	334
Table 671 – Associations of AbstractBuildingSubdivision (FeatureType)	334
Table 672 – Attributes of AbstractBuildingSubdivision (FeatureType)	335
Table 673 – Metadata of Building (TopLevelFeatureType)	335
Table 674 – Associations of Building (TopLevelFeatureType)	336
Table 675 – Attributes of Building (TopLevelFeatureType)	336
Table 676 – Metadata of BuildingConstructiveElement (FeatureType)	336
Table 677 – Attributes of BuildingConstructiveElement (FeatureType)	336
Table 678 – Metadata of BuildingFurniture (FeatureType)	337
Table 679 – Attributes of BuildingFurniture (FeatureType)	337
Table 680 – Metadata of BuildingInstallation (FeatureType)	337
Table 681 – Attributes of BuildingInstallation (FeatureType)	338
Table 682 – Metadata of BuildingPart (FeatureType)	338
Table 683 – Attributes of BuildingPart (FeatureType)	338
Table 684 – Metadata of BuildingRoom (FeatureType)	338
Table 685 – Associations of BuildingRoom (FeatureType)	339
Table 686 – Attributes of BuildingRoom (FeatureType)	339
Table 687 – Metadata of BuildingUnit (FeatureType)	339
Table 688 – Associations of BuildingUnit (FeatureType)	340
Table 689 – Attributes of BuildingUnit (FeatureType)	340
Table 690 – Metadata of Storey (FeatureType)	340
Table 691 – Associations of Storey (FeatureType)	340
Table 692 – Attributes of Storey (FeatureType)	341
Table 693 – Metadata of BuildingClassValue (CodeList)	341
Table 694 – Metadata of BuildingConstructiveElementClassValue (CodeList)	341
Table 695 – Metadata of BuildingConstructiveElementFunctionValue (CodeList)	342
Table 696 – Metadata of BuildingConstructiveElementUsageValue (CodeList)	342
Table 697 – Metadata of BuildingFunctionValue (CodeList)	342
Table 698 – Metadata of BuildingFurnitureClassValue (CodeList)	342
Table 699 – Metadata of BuildingFurnitureFunctionValue (CodeList)	343
Table 700 – Metadata of BuildingFurnitureUsageValue (CodeList)	343
Table 701 – Metadata of BuildingInstallationClassValue (CodeList)	343
Table 702 – Metadata of BuildingInstallationFunctionValue (CodeList)	344
Table 703 – Metadata of BuildingInstallationUsageValue (CodeList)	344
Table 704 – Metadata of BuildingRoomClassValue (CodeList)	344
Table 705 – Metadata of BuildingRoomFunctionValue (CodeList)	344
Table 706 – Metadata of BuildingRoomUsageValue (CodeList)	345
Table 707 – Metadata of BuildingSubdivisionClassValue (CodeList)	345
Table 708 – Metadata of BuildingSubdivisionFunctionValue (CodeList)	345

Table 709 – Metadata of BuildingSubdivisionUsageValue (CodeList)	345
Table 710 – Metadata of BuildingUsageValue (CodeList)	346
Table 711 – Metadata of RoofTypeValue (CodeList)	346
Table 712 – Metadata of RoomElevationReferenceValue (CodeList)	346
Table 713 – Metadata of ADEOfAbstractBuilding (DataType)	347
Table 714 – Metadata of ADEOfAbstractBuildingSubdivision (DataType)	347
Table 715 – Metadata of ADEOfBuilding (DataType)	347
Table 716 – Metadata of ADEOfBuildingConstructiveElement (DataType)	348
Table 717 – Metadata of ADEOfBuildingFurniture (DataType)	348
Table 718 – Metadata of ADEOfBuildingInstallation (DataType)	348
Table 719 – Metadata of ADEOfBuildingPart (DataType)	348
Table 720 – Metadata of ADEOfBuildingRoom (DataType)	349
Table 721 – Metadata of ADEOfBuildingUnit (DataType)	349
Table 722 – Metadata of ADEOfStorey (DataType)	349
Table 723 – Metadata of RoomHeight (DataType)	349
Table 724 – Attributes of RoomHeight (DataType)	350
Table 725 – Metadata of Tunnel (ApplicationSchema)	350
Table 726 – Metadata of AbstractTunnel (FeatureType)	351
Table 727 – Associations of AbstractTunnel (FeatureType)	351
Table 728 – Attributes of AbstractTunnel (FeatureType)	351
Table 729 – Metadata of HollowSpace (FeatureType)	352
Table 730 – Associations of HollowSpace (FeatureType)	352
Table 731 – Attributes of HollowSpace (FeatureType)	352
Table 732 – Metadata of Tunnel (TopLevelFeatureType)	352
Table 733 – Associations of Tunnel (TopLevelFeatureType)	353
Table 734 – Attributes of Tunnel (TopLevelFeatureType)	353
Table 735 – Metadata of TunnelConstructiveElement (FeatureType)	353
Table 736 – Attributes of TunnelConstructiveElement (FeatureType)	353
Table 737 – Metadata of TunnelFurniture (FeatureType)	354
Table 738 – Attributes of TunnelFurniture (FeatureType)	354
Table 739 – Metadata of TunnelInstallation (FeatureType)	354
Table 740 – Attributes of TunnelInstallation (FeatureType)	355
Table 741 – Metadata of TunnelPart (FeatureType)	355
Table 742 – Attributes of TunnelPart (FeatureType)	355
Table 743 – Metadata of HollowSpaceClassValue (CodeList)	356
Table 744 – Metadata of HollowSpaceFunctionValue (CodeList)	356
Table 745 – Metadata of HollowSpaceUsageValue (CodeList)	356
Table 746 – Metadata of TunnelClassValue (CodeList)	357
Table 747 – Metadata of TunnelConstructiveElementClassValue (CodeList)	357
Table 748 – Metadata of TunnelConstructiveElementFunctionValue (CodeList)	357
Table 749 – Metadata of TunnelConstructiveElementUsageValue (CodeList)	357

Table 750 – Metadata of TunnelFunctionValue (CodeList)	358
Table 751 – Metadata of TunnelFurnitureClassValue (CodeList)	358
Table 752 – Metadata of TunnelFurnitureFunctionValue (CodeList)	358
Table 753 – Metadata of TunnelFurnitureUsageValue (CodeList)	359
Table 754 – Metadata of TunnelInstallationClassValue (CodeList)	359
Table 755 – Metadata of TunnelInstallationFunctionValue (CodeList)	359
Table 756 – Metadata of TunnelInstallationUsageValue (CodeList)	359
Table 757 – Metadata of TunnelUsageValue (CodeList)	360
Table 758 – Metadata of ADEOfAbstractTunnel (DataType)	360
Table 759 – Metadata of ADEOfHollowSpace (DataType)	360
Table 760 – Metadata of ADEOfTunnel (DataType)	361
Table 761 – Metadata of ADEOfTunnelConstructiveElement (DataType)	361
Table 762 – Metadata of ADEOfTunnelFurniture (DataType)	361
Table 763 – Metadata of ADEOfTunnelInstallation (DataType)	361
Table 764 – Metadata of ADEOfTunnelPart (DataType)	362
Table C.1	416

LIST OF FIGURES

Figure 1 – UML notation (see ISO TS 19103, Geographic information – Conceptual schema language).	20
Figure 2 – Notation for class belonging to the Requirements Class which is subject of discussion in that clause	21
Figure 3 – Notation for class belonging to a Requirements Class different to that associated with the yellow color	22
Figure 4 – Notation for class defined in ISO 19107:2003, ISO 19111:2019, or ISO 19123:2005	22
Figure 5 – Notation for class defined in ISO 19109:2015	22
Figure 6 – Notation for UML notes and OCL constraints	22
Figure 7 – Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML Standard.	23
Figure 8 – CityGML 3.0 module overview. The vertical boxes show the different thematic modules. Horizontal modules specify concepts that are applicable to all thematic modules.	26
Figure 9 – Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).	30
Figure 10 – Examples of prototypic shapes (source: Rheinmetall Defence Electronics).	32
Figure 11 – Occupied and unoccupied spaces	34
Figure 12 – Representation of the same real-world building in the Levels of Detail 0-3.	36

Figure 13 – Floor plan representation (LOD0) of a building (left), combined LOD2 indoor and outdoor representation (right). Image adopted from Löwner et al. 2016.	37
Figure 14 – Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual ClosureSurface feature, which is both part of the DTM and the subsurface object (right) (graphic: IGG Uni Bonn).	38
Figure 15 – TerrainIntersectionCurve for a building (left, black) and a tunnel object (right, red). The tunnel's hollow space is sealed by a triangulated ClosureSurface (graphic: IGG Uni Bonn).	39
Figure 16 – Dynamizers link timeseries data coming from different sources to specific properties of individual city objects.	42
Figure 17 – CityGML UML Packages	47
Figure 18 – Use of ISO Standards in CityGML	51
Figure 19 – UML City Models and City Objects	53
Figure 20 – Representation of a carport as OccupiedSpace in different LODs. The red boxes represent solids, the green area represents a surface. In addition, the normal vectors of the roof solid (in red) and the roof surface (in green) are shown.	54
Figure 21 – Representation of a room as UnoccupiedSpace in different LODs. The red boxes represent solids, the green area represents a surface. In addition, the normal vectors of the room solid (in red) and the wall surface (in green) are shown.	55
Figure 22 – UML Space Concepts	55
Figure 23 – UML Geometry and LOD Concepts	58
Figure 24 – UML diagram of CityGML's core module	59
Figure 25 – ADE classes of the CityGML Core module.	60
Figure 26 – Basic Types, Enumerations, and Codelists from the CityGML Core module.	61
Figure 27 – UML diagram of CityGML's Appearance model.	66
Figure 28 – ADE classes of the CityGML Appearance Module.	67
Figure 29 – UML diagram of CityGML's City Furniture model.	70
Figure 30 – ADE classes and Code Lists of the CityGML CityFurniture module.	71
Figure 31 – UML diagram of the City Object Group Model.	74
Figure 32 – ADE classes of the CityGML CityObjectGroup module.	74
Figure 33 – Codelists from the CityGML CityObjectGroup module.	75
Figure 34 – UML diagram of the Dynamizer Model.	79
Figure 35 – ADE classes of the CityGML Dynamizer module.	79
Figure 36 – Codelists from the CityGML Dynamizer module.	80
Figure 37 – UML diagram of the Generics Model.	85
Figure 38 – ADE classes of the CityGML Generics module.	86
Figure 39 – Codelists from the CityGML Generics module.	86
Figure 40 – UML diagram of the Land Use Model.	91
Figure 41 – ADE classes of the CityGML Land Use module.	91
Figure 42 – Codelists from the CityGML Land Use module.	92
Figure 43 – UML diagram of the Point Cloud Model.	94
Figure 44 – ADE classes of the CityGML Point Cloud module.	95
Figure 45 – UML diagram of Relief module.	97

Figure 46 – ADE classes of the CityGML Relief module.	97
Figure 47 – UML diagram of the Transportation Model.	101
Figure 48 – ADE classes of the CityGML Transportation module.	102
Figure 49 – Codelists from the CityGML Transportation module.	103
Figure 50 – UML diagram of the Vegetation Model.	111
Figure 51 – ADE classes of the CityGML Vegetation module.	112
Figure 52 – Codelists from the CityGML Vegetation module.	112
Figure 53 – UML diagram of the Versioning Model.	116
Figure 54 – ADE classes of the CityGML Versioning module.	116
Figure 55 – UML diagram of the Water Body Model.	119
Figure 56 – ADE classes of the CityGML Water Body module.	119
Figure 57 – Codelists from the CityGML Water Body module.	120
Figure 58 – UML diagram of the Construction Model.	124
Figure 59 – ADE classes of the CityGML Construction module.	125
Figure 60 – Codelists from the CityGML Construction module.	125
Figure 61 – UML diagram of the Bridge Model.	133
Figure 62 – ADE classes of the CityGML Bridge module.	133
Figure 63 – Codelists from the CityGML Bridge module.	134
Figure 64 – UML diagram of CityGML's building model.	140
Figure 65 – ADE classes of the CityGML Building module.	141
Figure 66 – Codelists from the CityGML Building module.	141
Figure 67 – UML diagram of the Tunnel Model.	149
Figure 68 – ADE classes of the CityGML Tunnel module.	149
Figure 69 – Codelists from the CityGML Tunnel module.	150
Figure 70 – CityGML UML Packages	157
Figure 86 – The CityGML feature type Building is augmented with additional ADE properties by defining the data type EnergyProperties as a subclass of the ADE data type ADEOfBuilding.	367

LIST OF RECOMMENDATIONS

REQUIREMENTS CLASS 1	47
REQUIREMENTS CLASS 2	65
REQUIREMENTS CLASS 3	69
REQUIREMENTS CLASS 4	73
REQUIREMENTS CLASS 5	77
REQUIREMENTS CLASS 6	83
REQUIREMENTS CLASS 7	90

REQUIREMENTS CLASS 8	94
REQUIREMENTS CLASS 9	96
REQUIREMENTS CLASS 10	99
REQUIREMENTS CLASS 11	110
REQUIREMENTS CLASS 12	115
REQUIREMENTS CLASS 13	118
REQUIREMENTS CLASS 14	123
REQUIREMENTS CLASS 15	132
REQUIREMENTS CLASS 16	139
REQUIREMENTS CLASS 17	148
REQUIREMENTS CLASS 18	367
REQUIREMENT 1	49
REQUIREMENT 2	50
REQUIREMENT 3	50
REQUIREMENT 4	51
REQUIREMENT 5	67
REQUIREMENT 6	68
REQUIREMENT 7	71
REQUIREMENT 8	72
REQUIREMENT 9	72
REQUIREMENT 10	75
REQUIREMENT 11	76
REQUIREMENT 12	76
REQUIREMENT 13	80
REQUIREMENT 14	81
REQUIREMENT 15	87
REQUIREMENT 16	87
REQUIREMENT 17	88
REQUIREMENT 18	88
REQUIREMENT 19	92
REQUIREMENT 20	93
REQUIREMENT 21	95
REQUIREMENT 22	95

REQUIREMENT 23	98
REQUIREMENT 24	98
REQUIREMENT 25	104
REQUIREMENT 26	105
REQUIREMENT 27	105
REQUIREMENT 28	113
REQUIREMENT 29	113
REQUIREMENT 30	113
REQUIREMENT 31	116
REQUIREMENT 32	117
REQUIREMENT 33	120
REQUIREMENT 34	121
REQUIREMENT 35	121
REQUIREMENT 36	127
REQUIREMENT 37	128
REQUIREMENT 38	128
REQUIREMENT 39	136
REQUIREMENT 40	137
REQUIREMENT 41	137
REQUIREMENT 42	144
REQUIREMENT 43	144
REQUIREMENT 44	145
REQUIREMENT 45	152
REQUIREMENT 46	153
REQUIREMENT 47	153
REQUIREMENT 48	368
REQUIREMENT 49	368
REQUIREMENT 50	369
RECOMMENDATION 1	368
RECOMMENDATION 2	369
PERMISSION 1	50

ABSTRACT

This Standard defines the open CityGML Conceptual Model for the storage and exchange of virtual 3D city models. The CityGML Conceptual Model is defined by a Unified Modeling Language (UML) object model. This UML model builds on the ISO Technical Committee 211 (ISO/TC 211) conceptual model standards for spatial and temporal data. Building on the ISO foundation assures that the man-made features described in the city models share the same spatiotemporal universe as the surrounding countryside within which they reside.

A key goal for the development of the CityGML Conceptual Model is to provide 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.

The class models described in this standard are also available at <https://github.com/opengeospatial/CityGML3-Workspace/tree/1.0/UML/CityGML>

KEYWORDS

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

ogcdoc, OGC document, CityGML, 3D city models

SECURITY CONSIDERATIONS

No security considerations have been made for this document.

SUBMITTING ORGANIZATIONS

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

- Heazeltech LLC
- Institut national de l'information géographique et forestière (IGN)
- OpenSitePlan
- Technical University of Munich
- Ordnance Survey
- Virtual City Systems

SUBMITTERS

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

NAME	AFFILIATION
Emmanuel Devys	Institut national de l'information géographique et forestière (IGN), France
Volker Coors	HFT Stuttgart, Germany
Sylvester Hagler	U.S. National Geospatial-Intelligence Agency
Charles (Chuck) Heazel	HeazelTech LLC
Thomas H. Kolbe	Chair of Geoinformatics, Technical University of Munich, Germany
Tatjana Kutzner	Chair of Geoinformatics, Technical University of Munich, Germany
Claus Nagel	Virtual City Systems, Germany
Carsten Roensdorf	Ordnance Survey, Great Britain
Carl Stephen Smyth	OpenSitePlan, USA

PARTICIPANTS IN DEVELOPMENT

In addition to the Editors of the specification the following individuals contributed to the CityGML 3.0 development:

NAME	INSTITUTION
Giorgio Agugiaro	3D Geoinformation Group, Delft University of Technology, the Netherlands
Christof Beil	Chair of Geoinformatics, Technical University of Munich, Germany
Filip Biljecki	Department of Architecture, National University of Singapore, Singapore
Kanishk Chaturvedi	Chair of Geoinformatics, Technical University of Munich, Germany
Volker Coors	HFT Stuttgart, Germany
Emmanuel Devys	Institut national de l'information géographique et forestière (IGN), France
Jürgen Ebbinghaus	AED-SICAD, Germany
Heinrich Geerling	Architekturbüro Geerling, Germany
Gilles Gesquière	LIRIS, University of Lyon, France
Gerhard Gröger	CPA ReDev GmbH, Germany
Karl-Heinz Häfele	Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Germany
Nobuhiro Ishimaru	Hitachi, Ltd., Japan
Marc-Oliver Löwner	Institute for Geodesy and Photogrammetry, Technische Universität Braunschweig, Germany
Diana Moraru	Ordnance Survey, Great Britain
Friso Penninga	Geonovum, the Netherlands
Helga Tauscher	Faculty of Spatial Information, HTW Dresden – University of Applied Sciences, Germany
Linda van den Brink	Geonovum, the Netherlands

NAME	INSTITUTION
Heidi Vanparys	Danish Agency for Data Supply and Efficiency, Denmark
Sisi Zlatanova	Faculty of Built Environment, University of New South Wales, Australia

The table only lists persons that were involved in the development of CityGML 3.0. CityGML 3.0 is based on extensive previous work that was done for CityGML 2.0. For persons involved in the previous work, please refer to the CityGML 2.0 specification.

VII

INTRODUCTION

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 were defined as purely graphical or geometrical models, neglecting the semantic and topological aspects. Thus, these models could almost only be used for visualization purposes but not for thematic queries, analysis tasks, or spatial data mining. Since the limited reusability of models inhibits the broader use of 3D city models and may not justify the costs associated with maintaining city models, a more general modeling approach had to be taken in order to satisfy the information needs of the various application fields.

The CityGML Conceptual Model Standard defines 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 governments and companies to reap the benefits of their investment in 3D city models by being able to put the same models into play in 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.

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

CityGML enables the consistent representation of 3D urban objects across different geographic information systems and users.

VIII

ACKNOWLEDGEMENTS

The editors wish to thank the Special Interest Group 3D (SIG 3D) of the initiative Geodata Infrastructure Germany (GDI-DE) which originally started the development of CityGML, 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.



CityGML official logo

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>.

1

SCOPE

SCOPE

This Standard documents an OGC Conceptual Model (CM) Standard for specifying the representation of virtual 3D city and landscape models. The CityGML 3.0 Conceptual Model is a *Platform Independent Model (PIM)* (Clause 4.1.12). It defines concepts in a manner which is independent of any implementing technology. As such, the CityGML CM cannot be implemented directly. Rather, it serves as the base for *Platform Specific Models (PSM)* (Clause 4.1.13). A PSM adds to the PIM the technology-specific details needed to fully define the CityGML model for use with a specific technology. The PSM can then be used to generate the schema and other artifacts needed to build CityGML 3.0 implementations.

This Standard does not define the PSMs nor schemas for CityGML 3.0. Future CityGML 3.0 *Implementation Specifications (IS)* (Clause 4.1.8) will be developed to address this need. At a minimum, support for a Geography Markup Language (GML) Implementation Specification is expected. Additional Implementation Specifications for JSON and database schemas are also highly desirable.

A discussion of current and planned efforts to build Implementation Specifications for the CityGML Conceptual Model can be found in the [CityGML 3.0 Users Guide](#).

The target of the conformance classes specified in this document are:

- CityGML Implementation Specifications that provide encodings for the UML conceptual model specified in this document, and
- Additional UML models that can be created by users to extend this conceptual model as Application Domain Extensions (ADEs).

CityGML models are comprised of 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 that allows for the integration of a variety of source data to come together in a City Model. To enable the use of CityGML in specific domain areas, CityGML has historically provided an extension mechanism to enrich the data with identifiable features and properties, preserving semantic interoperability. Recognizing that an implementable expansion mechanism might have dependencies based on the encoding language, the CityGML 3.0 Conceptual Model specifies high level requirements rather than a full extension model.

Targeted application areas explicitly include:

- Urban and landscape planning;
- Architectural design;
- Tourist and leisure activities;
- Environmental, energy and mobility simulations;
- Mobile telecommunications;

- Disaster management;
- Homeland security;
- Vehicle and pedestrian navigation; and
- Training simulators and mobile robotics.

The future CityGML 3.0 Implementation Specifications will be implementable source formats for 3D portraying or transformation into dedicated portrayal formats such as the OGC I3S (OGC 17-014r7) or the OGC 3D Tiles Community Standards (OGC 18-053r2), OGC KML (OGC 12-007r2), Khronos COLLADA or Khronos glTF. The OGC 3D Portrayal Service (3DPS) (OGC 15-001r4) may be used for content delivery.

Features of the CityGML 3.0 Conceptual Model include the following.

- Geospatial Information Model (ontology) for urban landscapes based on the ISO 19100 family.
- Representation of 3D geometries, based on the ISO 19107 model, independent of data encodings, as well as of 3D point clouds.
- Grouping into space hierarchies, including concepts like stories/floors within buildings.
- Representation of object surface characteristics (e.g., textures, materials).
- Representation of dynamic, i.e., time-dependent, properties of city models.
- Taxonomies and aggregations including:
 - Digital Terrain Models as a combination of triangulated irregular networks (TINs), regular grids, break and skeleton lines, mass points;
 - Sites (currently buildings, other constructions, bridges, and tunnels);
 - Vegetation (areas, volumes, and solitary objects with vegetation classification);
 - Water bodies (volumes, surfaces);
 - Transportation facilities (graph structures, 3D space, 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; and
 - User-definable (recursive) grouping.

- Multiscale model with 4 well-defined consecutive Levels of Detail (LOD), applicable to both interior and exterior:
 - LOD0 – Highly generalized model;
 - LOD1 – Block model / extrusion objects;
 - LOD2 – Realistic, but still generalized model; and
 - LOD3 – Highly detailed model.
- Multiple representations in different LODs simultaneously and generalization relations between objects in different LODs.
- Ability to combine different interior and exterior LODs, including representation of floor plans.
- Optional topological connections between feature (sub)geometries.
- Enables a variety of different encoding specifications, including GML and JSON.
- Extension of the conceptual model through code lists, generic objects and Application Domain Extensions (ADEs).
- With CityGML 3.0, ADEs become platform-independent models on a conceptual level that can be mapped to multiple and different target encodings. ADEs are implemented as UML models that extend the conceptual model in this Standard. This includes a mechanism that favors the insertion of additional feature properties into any defined feature class through 'hooks' over subtyping of features. This means that the existing feature classes can be used and additional properties from one or more ADEs can easily be supported in different encodings.
- Ability to specify an ADE that can be further extended.



2

CONFORMANCE

CONFORMANCE

This Standard defines a **term Conceptual Model** not resolved via **ID Conceptual-Model** which is independent of any encoding or formatting techniques. The *Standardization Targets* (Annex B.1.1.14) for this Standard are:

1. *Conceptual Models* (Clause 4.1.6) (extended versions of this conceptual model)
2. *Implementation Specifications* (Clause 4.1.8) (encodings of this conceptual model)

2.1. Conceptual Models

A Conceptual Model standardization target is a version of the CityGML 3.0 Conceptual Model (CM) tailored for a specific user community. This tailoring can include:

1. Omission of one or more of the optional UML packages;
2. Reduction of the multiplicity for an attribute or association;
3. Restriction on the valid values for an attribute; and
4. Additional concepts documented through ADEs.

Of these options, actions #1, #2, and #3 can be performed when creating an implementation specification. Only action #4 requires an extension of the CityGML conceptual model. These extensions are accomplished using the ADE mechanism described in Application Domain Extensions (ADE) (Clause 9).

Extensions of the CityGML Conceptual Model conform with the ADE Conformance Class.

2.2. Implementation Specifications

Implementation Specifications define how a Conceptual Model should be implemented using a specific technology. Conformant Implementation Specifications provide evidence that they are an accurate representation of the Conceptual Model. This evidence should include implementations of the abstract tests specified in Annex A of this document.

Since this Standard is agnostic to the implementing technologies, the specific techniques to be used for conformance testing cannot be specified. Implementation Specifications need to provide evidence of conformance which is appropriate for the implementing technologies. This evidence should be provided as an annex to the Implementation Specification document.

2.3. Conformance Classes

This Standard identifies seventeen (17) conformance classes. One conformance class is defined for each package in the UML model. Each conformance class is defined by one requirements class. The tests in Annex A are organized by Requirements Class. So an implementation of the *Core* conformance class must pass all tests specified in Annex A for the *Core* requirements class.

Of these seventeen conformance classes, only the *Core* conformance class is mandatory. All other conformance classes are optional. In the case where a conformance class has a dependency on another conformance class, that conformance class should also be implemented.

The CityGML Conceptual Model is defined by the CityGML UML model. This Standard is a representation of that UML model in document form. In the case of a discrepancy between the UML model and this document, the UML model takes precedence.

3

NORMATIVE REFERENCES

NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Cliff Kottman and Carl Reed: OGC 08-126, *Topic 5 – Features*. Open Geospatial Consortium (2009). https://portal.ogc.org/files/?artifact_id=29536.

Cliff Kottman: OGC 99-108r2, *Topic 8 – Relationships Between Features*. Open Geospatial Consortium (1999). https://portal.ogc.org/files/?artifact_id=894.

Cliff Kottman: OGC 99-110, *Topic 10 – Feature Collections*. Open Geospatial Consortium (1999). https://portal.ogc.org/files/?artifact_id=897.

Alexandre Robin: OGC 08-094r1, *OGC® SWE Common Data Model Encoding Standard*. Open Geospatial Consortium (2011). https://portal.ogc.org/files/?artifact_id=41157.

James Tomkins , Dominic Lowe: OGC 15-043r3, *Timeseries Profile of Observations and Measurements* . Open Geospatial Consortium (2016). <https://docs.ogc.org/is/15-043r3/15-043r3.html>.

ISO: ISO 19101-1:2014, *Geographic information – Reference model – Part 1: Fundamentals*. International Organization for Standardization, Geneva (2014). <https://www.iso.org/standard/59164.html>.

ISO: ISO 19101-2:2018, *Geographic information – Reference model – Part 2: Imagery*. International Organization for Standardization, Geneva (2018). <https://www.iso.org/standard/69325.html>.

ISO: ISO 19103:2015, *Geographic information – Conceptual schema language*. International Organization for Standardization, Geneva (2015). <https://www.iso.org/standard/56734.html>.

ISO: ISO 19105:2000, *Geographic information – Conformance and testing*. International Organization for Standardization, Geneva (2000). <https://www.iso.org/standard/26010.html>.

ISO: ISO 19107:2003, *Geographic information – Spatial schema*. International Organization for Standardization, Geneva (2003). <https://www.iso.org/standard/26012.html>.

ISO: ISO 19108:2006, ISO: ISO 19108:2002/Cor 1:2006, *Geographic information – Temporal schema – Technical Corrigendum 1* (2006). ISO (2006).

ISO: ISO 19109:2015, *Geographic information – Rules for application schema*. International Organization for Standardization, Geneva (2015). <https://www.iso.org/standard/59193.html>.

ISO: ISO 19111:2019, *Geographic information – Referencing by coordinates*. International Organization for Standardization, Geneva (2019). <https://www.iso.org/standard/74039.html>.

ISO: ISO 19123:2005, *Geographic information – Schema for coverage geometry and functions*. International Organization for Standardization, Geneva (2005). <https://www.iso.org/standard/40121.html>.

ISO: ISO 19143:2010, *Geographic information – Filter encoding*. International Organization for Standardization, Geneva (2010). <https://www.iso.org/standard/42137.html>.

ISO: ISO 19156:2011, *Geographic information – Observations and measurements*. International Organization for Standardization, Geneva (2011). <https://www.iso.org/standard/32574.html>.

ISO/IEC: ISO/IEC 19505-2:2012, *Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2012). <https://www.iso.org/standard/52854.html>.

ISO/IEC: ISO/IEC 19507:2012, *Information technology – Object Management Group Object Constraint Language (OCL)*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2012). <https://www.iso.org/standard/57306.html>.

ISO/IEC: ISO/IEC 19775-1:2013, *Information technology – Computer graphics, image processing and environmental data representation – Extensible 3D (X3D) – Part 1: Architecture and base components*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2013). <https://www.iso.org/standard/60760.html>.

N. Freed, N. Borenstein: IETF RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. (1996). <https://www.rfc-editor.org/info/rfc2045>.

N. Freed, N. Borenstein: IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. (1996). <https://www.rfc-editor.org/info/rfc2046>.

T. Berners-Lee, R. Fielding, L. Masinter: IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. (2005). <https://www.rfc-editor.org/info/rfc3986>.

INSPIRE: D2.8.III.2 Data Specification on Buildings – Technical Guidelines. European Commission Joint Research Centre. (2013)

Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.5.0 (2008)

OASIS: Customer Information Quality Specifications – extensible Address Language (xAL), Version v3.0 (2008)

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in [OGC Policy Directive 49](#), which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications ([OGC 08-131r3](#)), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

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

4.1. Terms and definitions

4.1.1. 2D data

geometry of features is represented in a two-dimensional space

Note 1 to entry: In other words, the geometry of 2D data is given using (X,Y) coordinates.

[**SOURCE:** INSPIRE: D2.8.III.2, Definition 1]

4.1.2. 2.5D data

geometry of features is represented in a three-dimensional space with the constraint that, for each (X,Y) position, there is only one Z

[**SOURCE:** INSPIRE: D2.8.III.2, Definition 2]

4.1.3. 3D data

Geometry of features is represented in a three-dimensional space.

[SOURCE: INSPIRE: D2.8.III.2, Definition 3]

4.1.4. application schema

A set of *conceptual schema* (Clause 4.1.7) for data required by one or more applications. An application schema contains selected parts of the base schemas presented in the ORM Information Viewpoint. Designers of application schemas may extend or restrict the types defined in the base schemas to define appropriate types for an application domain. Application schemas are information models for a specific information community.

[SOURCE: OGC Definitions Register, URI <http://www.opengis.net/def/glossary/term/ApplicationSchema>]

4.1.5. codelist

A value domain including a code for each permissible value.

4.1.6. conceptual model

model that defines concepts of a universe of discourse

[SOURCE: ISO 19101-1:2014, Clause 4.1.5]

4.1.7. conceptual schema

- a) formal description of a *conceptual model* (Clause 4.1.6)

[SOURCE: ISO 19101-1:2014, Clause 4.1.6]

- b) base schema. Formal description of the model of any geospatial information. *Application schemas* (Clause 4.1.4) are built from conceptual schemas.

[**SOURCE:** OGC Definitions Register, URI <http://www.opengis.net/def/glossary/term/ConceptualSchema>]

4.1.8. Implementation Specification

Specified on the OGC Document Types Register

[**SOURCE:** OGC Document Types Register, URI <http://www.opengis.net/def/doc-type/is>]

4.1.9. levels of detail

quantity of information that portrays the real world

Note 1 to entry: The concept comprises data capturing rules of spatial object types, the accuracy and the types of geometries, and other aspects of a data specification. In particular, it is related to the notions of scale and resolution.

[**SOURCE:** INSPIRE Glossary]

4.1.10. life-cycle information

set of properties of a spatial object that describe the temporal characteristics of a version of a spatial object or the changes between versions

[**SOURCE:** INSPIRE Glossary]

4.1.11. Platform (Model Driven Architecture)

the set of resources on which a system is realized.

[**SOURCE:** OMG MDA]

4.1.12. Platform Independent Model

a model that is independent of a specific platform

[SOURCE: OMG MDA]

4.1.13. Platform Specific Model

a model of a system that is defined in terms of a specific platform

[SOURCE: OMG MDA]

4.2. Abbreviated terms

AEC	Architecture, Engineering, Construction
ALKIS	German National Standard for Cadastral Information
ATKIS	German National Standard for Topographic and Cartographic Information
B-Rep	Boundary Representation
BIM	Building Information Modeling
bSI	buildingSMART International
CAD	Computer Aided Design
COLLADA	Collaborative Design Activity
CSG	Constructive Solid Geometry
DTM	Digital Terrain Model
DXF	Drawing Exchange Format
ESRI	Environmental Systems Research Institute
EuroSDR	European Spatial Data Research Organisation
FM	Facility Management
GDF	Geographic Data Files

GDI	NRW Geodata Infrastructure North-Rhine Westphalia
GDI-DE	Spatial Data Infrastructure Germany (Geodateninfrastruktur Deutschland)
GML	Geography Markup Language
IAI	International Alliance for Interoperability (now buildingSMART International (bSI))
IETF	Internet Engineering Task Force
IFC	Industry Foundation Classes
IoT	Internet of Things
ISO	International Organization for Standardisation
ISO/TC211	ISO Technical Committee 211
LOD	Levels of Detail
MQTT	Message Queuing Telemetry Transport
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
SIG	Special Interest Group 3D of the GDI-DE
TIC	Terrain Intersection Curve
TIN	Triangulated Irregular Network
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VRML	Virtual Reality Modeling Language
WFS	OGC Web Feature Service
W3C	World Wide Web Consortium
W3DS	OGC Web 3D Service
xAL	OASIS extensible Address Language
XML	Extensible Markup Language
X3D	Open Standards XML-enabled 3D file format of the Web 3D Consortium

2D	Two Dimensional
3D	Three Dimensional



5

CONVENTIONS

5.1. Identifiers

The normative provisions in this document are denoted by the URI

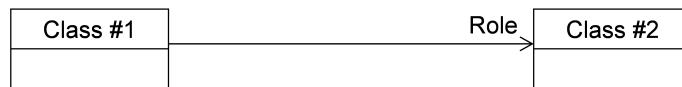
<http://www.opengis.net/spec/CityGML-1/3.0>

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

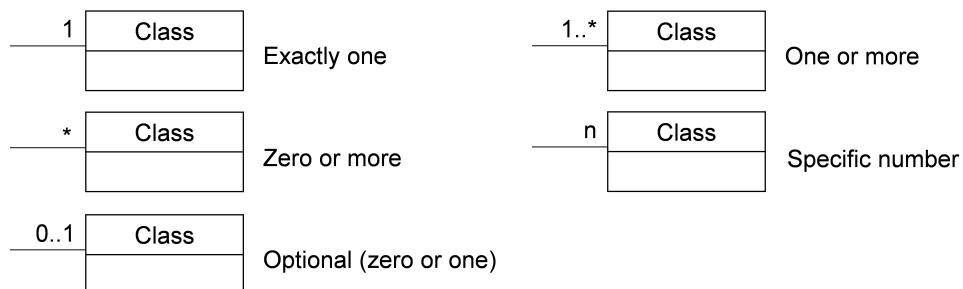
5.2. UML Notation

The CityGML Conceptual Model (CM) Standard is presented in this document through 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 in Figure 1.

Association between classes



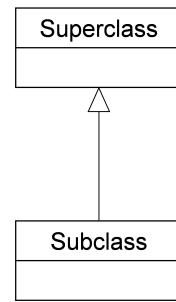
Association cardinality



Aggregation between classes



Class inheritance



Composition between classes



Figure 1 – UML notation (see ISO TS 19103, Geographic information – Conceptual schema language).

All associations between model elements in the CityGML Conceptual Model are uni-directional. Thus, associations in the model 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 in this model.

- «ApplicationSchema» denotes a conceptual schema for data required by one or more applications. In the CityGML Conceptual Model, every module is defined as a separate application schema to allow for modularization.
- «FeatureType» represents features that are similar and exhibit common characteristics. Features are abstractions of real-world phenomena and have an identity.

- «TopLevelFeatureType» denotes features that represent the main components of the conceptual model. Top-level features may be further semantically and spatially decomposed and substructured into parts.
- «Type» denotes classes that are not directly instantiable, but are used as an abstract collection of operation, attribute and relation signatures. The stereotype is used in the CityGML Conceptual Model only for classes that are imported from the ISO standards 19107, 19109, 19111, and 19123.
- «ObjectType» represents objects that have an identity, but are not features.
- «DataType» defines a set of properties that lack identity. A data type is a classifier with no operations, whose primary purpose is to hold information.
- «Enumeration» enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified in the CityGML Conceptual Model.
- «BasicType» defines a basic data type.
- «CodeList» enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline in the CityGML UML Model. The allowed values can be provided within an external code list.
- «Union» is a list of attributes. The semantics are that only one of the attributes can be present at any time.
- «Property» denotes attributes and association roles. This stereotype does not add further semantics to the conceptual model, but is required to be able to add tagged values to the attributes and association roles that are relevant for the encoding.
- «Version» denotes that the value of an association role that ends at a feature type is a specific version of the feature, not the feature in general.

In order to enhance the readability of the CityGML UML diagrams, classes are depicted in different colors. The following coloring scheme is applied.

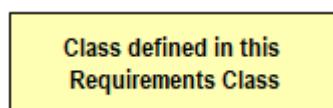
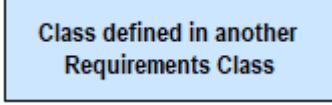


Figure 2 – Notation for class belonging to the Requirements Class which is subject of discussion in that clause

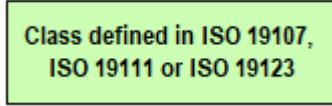
Classes painted in yellow belong to the Requirements Class which is subject of discussion in that clause of the standard in which the UML diagram is given. For example, in the context of Clause 7.2, which introduces the CityGML Core module, the yellow color is used to denote classes that are defined in the CityGML Core Requirements Class. Likewise, the yellow classes shown in the UML diagram in Clause 7.17 are associated with the *Building Requirements Class* that is subject of discussion in that chapter.



Class defined in another Requirements Class

Figure 3 – Notation for class belonging to a Requirements Class different to that associated with the yellow color

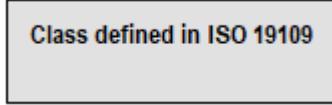
Classes painted in blue belong to a Requirements Class different to that associated with the yellow color. In order to explicitly denote to which Requirements Class these classes belong, their class names are preceded by the UML package name of that Requirements Class. For example, in the context of the *Building Requirements Class*, classes from the *CityGML Core* and the *Construction Requirements Classes* are painted in blue and their class names are preceded by *Core* and *Construction*, respectively.



Class defined in ISO 19107, ISO 19111 or ISO 19123

Figure 4 – Notation for class defined in ISO 19107:2003, ISO 19111:2019, or ISO 19123:2005

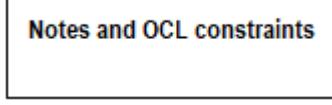
Classes painted in green are defined in the ISO standards ISO 19107:2003, ISO 19111:2019, or ISO 19123:2005. Their class names are preceded by the UML package name, in which the classes are defined.



Class defined in ISO 19109

Figure 5 – Notation for class defined in ISO 19109:2015

Classes painted in grey are defined in the ISO standard ISO 19109:2015. In the context of this Standard, this only applies to the class *AnyFeature*. *AnyFeature* is an instance of the metaclass *FeatureType* and acts as super class of all classes in the CityGML UML model with the stereotype «*FeatureType*». A metaclass is a class whose instances are classes.



Notes and OCL constraints

Figure 6 – Notation for UML notes and OCL constraints

The color white is used for notes and Object Constraint Language (OCL) (ISO/IEC 19507:2012) constraints that are provided in the UML diagrams.

The example UML diagram in Figure 7 demonstrates the UML notation and coloring scheme used throughout this Standard. In this example, the yellow classes are associated with the *CityGML Building* module, the blue classes are from the *CityGML Core* and *Construction* modules, and the green class depicts a geometry element defined by ISO 19107:2003.

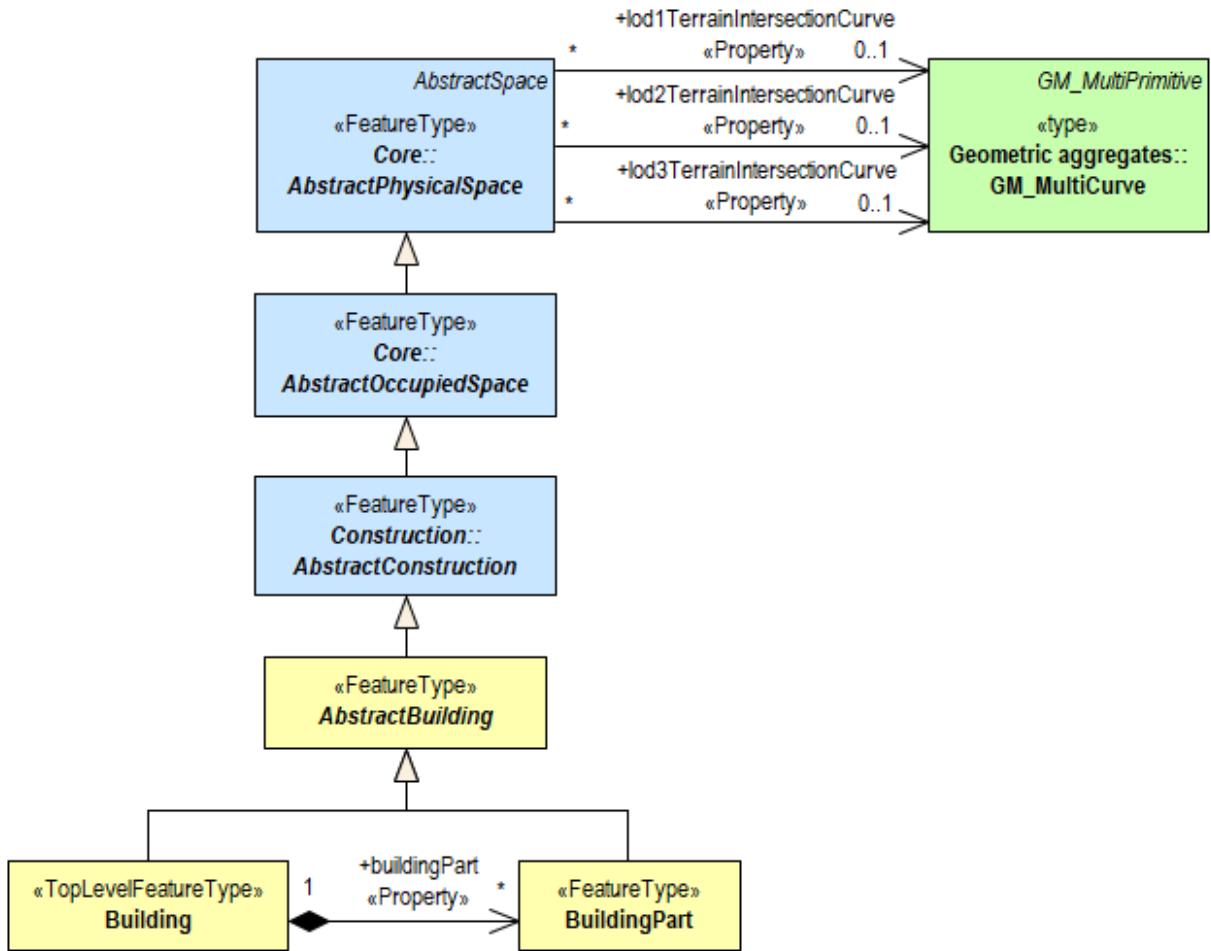


Figure 7 – Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML Standard.

The UML diagrams presented in this standard can also be found in XML Metadata Interchange (XMI) format at <https://github.com/opengeospatial/CityGML3-Workspace/tree/1.0/UML/CityGML>

6

OVERVIEW OF CITYGML

This Standard defines an open CityGML Conceptual Model (CM) for the storage and exchange of virtual 3D city and landscape models. These models include the most relevant entities of the urban space like buildings, roads, railways, tunnels, bridges, city furniture, water bodies, vegetation, and the terrain. The conceptual schema specifies how and into which parts and pieces physical objects of the real world should be decomposed and classified. All objects can be represented with respect to their semantics, 3D geometry, 3D topology, appearances, and their changes over time. Different spatial representations can be provided for each object (outdoor and indoor) in four predefined Levels of Detail (LOD 0-3). The CityGML 3.0 Conceptual Model (Clause 7) is formally specified using UML class diagrams, complemented by a data dictionary (Clause 8) providing the definitions and explanations of the object classes and attributes. This Conceptual Model is the basis for multiple encoding standards, which map the concepts (or subsets thereof) onto exchange formats or database structures for data exchange and storage.

While the CityGML Conceptual Model can be used for 3D visualization purposes, its special merits lie in applications that go beyond visualization such as decision support, urban and landscape planning, urban facility management, Smart Cities, navigation (both indoor and outdoor), Building Information Modeling (especially for as-built documentation), integration of city and BIM models, assisted and autonomous driving, and simulations in general (cf. Kolbe 2009). A comprehensive overview on the many different applications of virtual 3D city models is given in [Biljecki et al. 2015]. Many of the applications already use and some even require using CityGML.

In the CityGML CM, all 3D city objects can easily be enriched with thematic data. For example, street objects can be enriched with information about traffic density, speed limit, number of lanes etc., or buildings can be enriched by information on the heating and electrical energy demand, numbers of households and inhabitants, the appraised building value etc. Even building parts such as individual roof or wall surfaces can be enriched with information e.g., about solar irradiation and thermal insulation parameters. For many application domains specific extensions of the CityGML CM have already been created (cf. Biljecki et al. 2018).

6.1. Modularization

The CityGML Conceptual Model provides models for the most important types of objects within virtual 3D city and landscape models. These feature types have been identified to be either required or important in many different application areas. However, implementations are not required to support the complete CityGML model in order to be conformant to the standard. Implementations may employ a subset of constructs according to their specific information needs. For this purpose, modularization is applied to the CityGML CM.

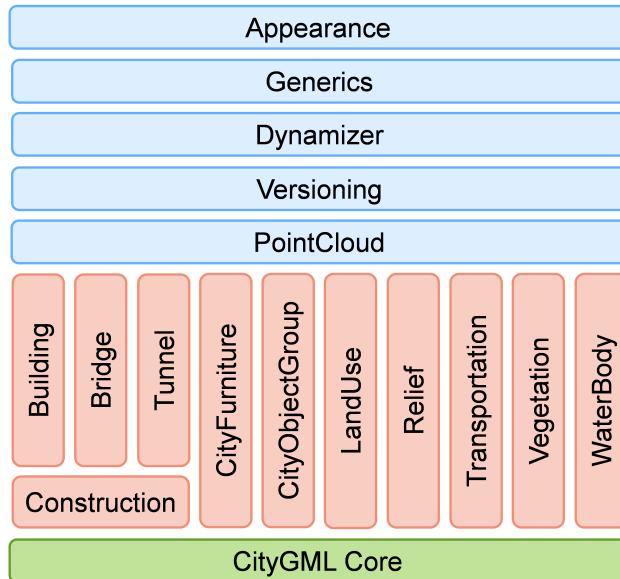


Figure 8 – CityGML 3.0 module overview. The vertical boxes show the different thematic modules. Horizontal modules specify concepts that are applicable to all thematic modules.

The CityGML conceptual model is thematically decomposed into a *Core module* and different kinds of *extension modules* as shown in Figure 8. The Core module (shown in green) comprises the basic concepts and components of the CityGML CM and, thus, must be implemented by any conformant system. Each red colored module covers a specific thematic field of virtual 3D city models.

The CityGML CM introduces the following eleven thematic extension modules: *Building*, *Bridge*, *Tunnel*, *Construction*, *CityFurniture*, *CityObjectGroup*, *LandUse*, *Relief*, *Transportation*, *Vegetation*, and *WaterBody*. All three modules *Building*, *Bridge*, and *Tunnel* model civil structures and share common concepts that are grouped within the *Construction* module. The five blue colored extension modules add specific modeling aspects that can be used in conjunction with all thematic modules.

- The *Appearance* module contains the concepts to represent appearances (like textures and colors) of city objects.
- The *PointCloud* module provides concepts to represent the geometry of city objects by 3D point clouds.
- The *Generics* module defines the concepts for generic objects, attributes, and relationships.
- *Versioning* adds concepts for the representation of concurrent versions, real world object histories and feature histories.
- The *Dynamizer* module contains the concepts to represent city object properties by time series data and to link them with sensors, sensor data services or external files.

Each CityGML encoding can specify support for a subset of the CityGML modules only. If a module is supported by an encoding, then all concepts should be mapped. However, the

encoding specification can define so-called *null mappings* to restrict the use of specific elements of the conceptual model in an encoding. Null mappings can be expressed in an encoding specification for individual feature types, properties, and associations defined within a CityGML module. This means that the corresponding element will not be included in the respective encoding.

Note that also CityGML applications do not have to support all modules. Applications can also decide to only support a specific subset of CityGML modules. For example, when an application only has to work with building data, only the modules *Core*, *Construction*, and *Building* would have to be supported.

6.2. General Modeling Principles

6.2.1. Semantic Modeling of Real-World Objects

Real-world objects are represented by geographic features according to the definition in ISO 19109. Geographic features of the same type (e.g., buildings, roads) are modeled by corresponding feature types that are represented as classes in the Conceptual Model (CM). The objects within a 3D city model are instances of the different feature types.

In order to distinguish and reference individual objects, each object has unique identifiers. In the CityGML 3.0 CM, each geographic feature has the mandatory *featureID* and an optional *identifier* property. The *featureID* is used to distinguish all objects and possible multiple versions of the same real-world object. The *identifier* is identical for all versions of the same real-world object and can be used to reference specific objects independent from their actual object version. The *featureID* is unique within the same CityGML dataset, but it is generally recommended to use globally unique identifiers like UUID values or identifiers maintained by an organization such as a mapping agency. Providing globally unique and stable identifiers for the *identifier* attribute is recommended. This means these identifiers should remain stable over the lifetime of the real-world object.

CityGML feature types typically have a number of spatial and non-spatial properties (also called attributes) as well as relationships with other feature or object types. Note that a single CityGML object can have different spatial representations at the same time. For example, different geometry objects representing the feature's geometry in different levels of detail or as different spatial abstractions.

Many attributes have simple, scalar values like a number or a character string. However, some attributes are complex. They do not just have a single property value. In CityGML the following types of complex attributes occur.

- *Qualified attribute values*: For example, a measure consists of the value and a reference to the unit of measure, or e.g., for relative and absolute height levels the reference level has to also be named.

- *Code list values*: A code list is a form of enumeration where the valid values are defined in a separate register. The code list values consist of a link or identifier for the register as well as the value from that register which is being used.
- Attributes consisting of a *tuple of different fields and values*: For example, addresses, space occupancy, and others.
- Attribute value consisting of a *list of numbers*: For example, representing coordinate lists or matrices.

In order to support feature history, CityGML 3.0 introduces bitemporal timestamps for all objects. In CityGML 2.0, the attributes *creationDate* and *terminationDate* are supported. These refer to the time period in which a specific version of an object is an integral part of the 3D city model. In 3.0, all features can now additionally have the attributes *validFrom* and *validTo*. These represent the lifespan a specific version of an object has in the real-world. Using these two time intervals a CityGML dataset could be queried both for how did the city look alike at a specific point in time as well as how did the *city model* look at that time.

The combination of the two types of feature identifiers and bitemporal timestamps enables encoding not only the current version of a 3D city model, but also the model's entire history can be represented in CityGML and possibly exchanged within a single file.

6.2.2. Class Hierarchy and Inheritance of Properties and Relations

In CityGML, the specific feature types like *Building*, *Tunnel*, or *WaterBody* are defined as subclasses of more general higher-level classes. Hence, feature types build a hierarchy along specialization / generalization relationships where more specialized feature types inherit the properties and relationships of all their superclasses along the entire generalization path to the topmost feature type *AnyFeature*.

NOTE: A superclass is the class from which subclasses can be created.

6.2.3. Relationships between CityGML objects

In CityGML, objects can be related to each other and different types of relations are distinguished. First of all, complex objects like buildings or transportation objects typically consist of parts. These parts are individual features of their own, and can even be further decomposed. Therefore, CityGML objects can form aggregation hierarchies. Some feature types are marked in the conceptual model with the stereotype «*TopLevelFeatureType*». These constitute the main objects of a city model and are typically the root of an aggregation hierarchy. Only top-level features are allowed as direct members of a city model. The information about which feature types belong to the top level is required for software packages that want to filter imports, exports, and visualizations according to the general type of a city object (e.g., only show buildings, solitary vegetation objects, and roads). CityGML Application Domain Extensions should also make use of this concept, such that software tools can learn from inspecting their conceptual schema what are the main, i.e., the top-level, feature types of the extension.

Some relations in CityGML are qualified by additional parameters, typically to further specify the type of relationship. For example, a relationship can be qualified with a URI pointing to a definition of the respective relation type in an Ontology. Qualified relationships are used in CityGML, among others, for:

- General relationships between features – association *relatedTo* between city objects,
- User-defined aggregations using *CityObjectGroup*. This relation allows also for recursive aggregations,
- External references – linking of city objects with corresponding entities from external resources like objects in a cadastre or within a BIM dataset.

The CityGML CM contains many relationships that are specifically defined between certain feature types. For example, there is the *boundary* relationship from 3D volumetric objects to its thematically differentiated 3D boundary surfaces. Another example is the *generalizesTo* relation between feature instances that represent objects on different generalization levels.

In the CityGML 3.0 CM there are new associations to express topologic, geometric, and semantic relations between all kinds of city objects. For example, information that two rooms are adjacent or that one interior building installation (like a curtain rail) is overlapping with the spaces of two connected rooms can be expressed. The CM also enables documenting that two wall surfaces are parallel and two others are orthogonal. Also distances between objects can be represented explicitly using geometric relations. In addition to spatial relations logical relations can be expressed.

6.2.4. Definition of the Semantics for all Classes, Properties, and Relations

The meanings of all elements defined in the CityGML conceptual model are normatively specified in the data dictionary in Clause 8.

6.3. Representation of Spatial Properties

6.3.1. Geometry and Topology

Spatial properties of all CityGML feature types are represented using the geometry classes defined in ISO 19107. Spatial representations can have 0-, 1-, 2-, or 3-dimensional extents depending on the respective feature type and Levels of Detail (LOD; the LOD concept is discussed in Clause 6.4.4 and Clause 7.2.5). With only a few exceptions, all geometries must use 3D coordinate values. Besides primitive geometries like single points, curves, surfaces, and solids, CityGML makes use of different kinds of aggregations of geometries like spatial aggregates (*MultiPoint*, *MultiCurve*, *MultiSurface*, *MultiSolid*) and composites (*CompositeCurve*, *CompositeSurface*, *CompositeSolid*). Volumetric shapes are represented in ISO 19107 according to the so-called *Boundary Representation* (B-Rep, for explanation see Foley et al. 2002) only.

The CityGML Conceptual Model does not put any restriction on the usage of specific geometry types as defined in ISO 19107. For example, 3D surfaces could be represented in a dataset using 3D polygons or 3D meshes such as triangulated irregular networks (TINS) or by non-uniform rational B-spline surfaces (NURBS). However, an encoding may restrict the usage of geometry types. For example, curved lines like B-splines or clothoids, or curved surfaces like NURBS could be disallowed by explicitly defining *null encodings* for these concepts in the encoding specification (c.f. Clause 6.1 above).

Note that the conceptual schema of ISO 19107 allows composite geometries to be defined by a recursive aggregation 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*). This is illustrated in Figure 9.

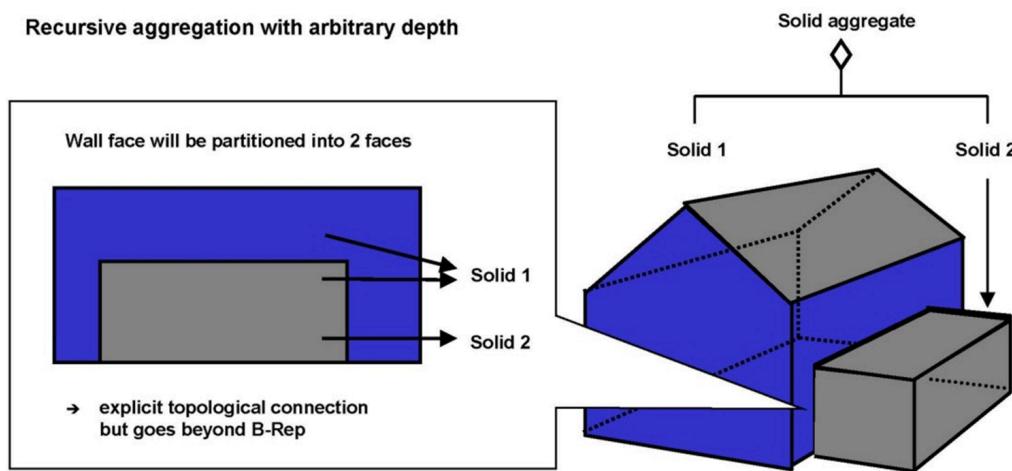


Figure 9 – Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

While the CityGML Conceptual Model does not employ the topology classes from ISO 19107, topological relations between geometries can be established by sharing geometries (typically parts of the boundary) between different geometric objects. One part of real-world space can be 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 can be avoided and explicit topological relations between parts are maintained.

Basically, there are three cases for sharing geometries.

- First, two different semantic objects may be spatially represented by the same geometry object. For example, if a foot path is both a transportation feature and a vegetation feature, the surface geometry defining the path is referenced by both the transportation object and by the vegetation object.
- Second, a geometry object may be shared between a feature and another geometry. For example, 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 Figure 9, this requires partitioning of the respective surfaces.

In general, B-Rep 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.

Another example of sharing geometry objects that are members of the boundaries in different higher-dimensional geometry objects is the sharing of point geometries or curve geometries, which make up the outer and inner boundaries of a polygon. This means that each point is only represented once, and different polygons could reference this point geometry. The same applies to the representation of curves for transportation objects like roads, whose end points could be shared such as between different road segments to topologically connect them.

Note that the use of topology in CityGML datasets by sharing geometries is optional. Furthermore, an encoding of the CityGML conceptual model might restrict the usage of shared geometries. For example, it might only be allowed to share identical (support) points from different 3D polygons or only entire polygons can be shared between touching solids (like shown in Figure 9).

6.3.2. 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 (see Figure 10). 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 (CRS) 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. Since the ISO 19107 geometry model does not provide support for scene graph concepts, the CityGML class `ImplicitGeometry` has been introduced (for further description see Clause 7.2.5). The prototype geometry can be represented using ISO 19107 geometry objects or by referencing an external file containing the geometry in another data format.

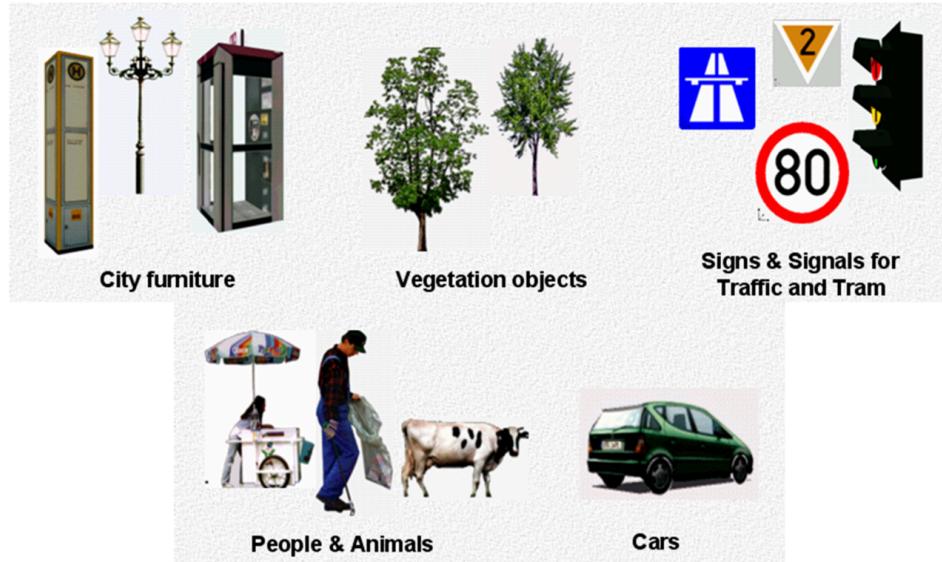


Figure 10 – Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

6.3.3. Point Cloud Representation

In addition to the spatial representations defined in the Core module, the geometry of physical spaces and of thematic surfaces can now also be provided by 3D point clouds using MultiPoint geometry. This allows, for example, spatially representing the building hull, a room within a building or a single wall surface just by a point cloud. All thematic feature types including transportation objects, vegetation, city furniture, etc. can also be spatially represented by point clouds. In this way, the ClearanceSpace of a road or railway could, for instance, be modeled directly from the result of a mobile laser scanning campaign. Point clouds can either be included in a CityGML dataset or just reference an external file of some common types such as LAS or LAZ.

6.3.4. Coordinate Reference Systems (CRS)

CityGML is about 3D city and landscape models. This means that nearly all geometries use 3D coordinates, where each single point and also the points defining the boundaries of surfaces and solids have three coordinate values (x,y,z) each. Coordinates always have to be given with respect to a coordinate reference system (CRS) that relates them unambiguously with a specific position on the Earth. In contrast to CAD or BIM, each 3D point is absolutely georeferenced, which makes CityGML especially suitable to represent geographically large extended structures like airports, railways, bridges, dams, where the Earth curvature has a significant effect on the object's geometry (for further explanations see Kaden Clemen 2017).

In most CRS, the (x,y) coordinates refer to the horizontal position of a point on the Earth's surface. The z coordinate typically refers to the vertical height over (or under) the reference surface. Note that depending on the chosen CRS, x and y may be given as angular values like latitude and longitude or as distance values in meters or feet. According to ISO 19111,

numerous 3D CRS can be used. This includes global as well as national reference systems using geocentric, geodetic, or projected coordinate systems.

6.4. CityGML Core Model: Space Concept, Levels of Detail, Special Spatial Types

6.4.1. Spaces and Space Boundaries

In the CityGML 3.0 Conceptual Model, a clear semantic distinction of spatial features is introduced by mapping all city objects onto the semantic concepts of spaces and space boundaries. A *Space* is an entity of volumetric extent in the real world. Buildings, water bodies, trees, rooms, and traffic spaces are examples for such entities with volumetric extent. A *Space Boundary* is an entity with areal extent in the real world. Space Boundaries delimit and connect Spaces. Examples are the wall surfaces and roof surfaces that bound a building, the water surface as boundary between the water body and air, the road surface as boundary between the ground and the traffic space, or the digital terrain model representing the space boundary between the over- and underground space.

To obtain a more precise definition of spaces, they are further subdivided into physical spaces and logical spaces. Physical spaces are spaces that are fully or partially bounded by physical objects. Buildings and rooms, for instance, are physical spaces as they are bounded by walls and slabs. Traffic spaces of roads are physical spaces as they are bounded by road surfaces against the ground. Logical spaces, in contrast, are spaces that are not necessarily bounded by physical objects, but are defined according to thematic considerations. Depending on the application, logical spaces can also be bounded by non-physical, i.e., virtual boundaries, and they can represent aggregations of physical spaces. A building unit, for instance, is a logical space as it aggregates specific rooms to flats, the rooms being the physical spaces that are bounded by wall surfaces, whereas the aggregation as a whole is being delimited by a virtual boundary. Other examples are city districts which are bounded by virtual vertically extruded administrative boundaries, public spaces vs. Security zones in airports, or city zones with specific regulations stemming from urban planning. The definition of physical and logical spaces and of corresponding physical and virtual boundaries is in line with the discussion in [Smith Varzi 2000] on the difference between bona fide and fiat boundaries to bound objects. Bona fide boundaries are physical boundaries; they correspond to the physical boundaries of physical spaces in the CityGML 3.0 CM. In contrast, fiat boundaries are man-made boundaries. They are equivalent to the virtual boundaries of logical spaces.

Physical spaces, in turn, are further classified into occupied spaces and unoccupied spaces. Occupied spaces represent physical volumetric objects that occupy space in the urban environment. Examples for occupied spaces are buildings, bridges, trees, city furniture, and water bodies. Occupying space means that some space is blocked by these volumetric objects. For instance, the space blocked by the building in Figure 11 cannot be used any more for driving through this space or placing a tree on that space. In contrast, unoccupied spaces represent physical volumetric entities that do not occupy space in the urban environment, i.e., no space

is blocked by these volumetric objects. Examples for unoccupied spaces are building rooms and traffic spaces. There is a risk of misunderstanding the term *OccupiedSpace*. However, we decided to use the term anyway, as it is established in the field of robotics for over three decades [Elfes 1989]. The navigation of mobile robots makes use of a so-called occupancy map that marks areas that are occupied by matter and, thus, are not navigable for robots.

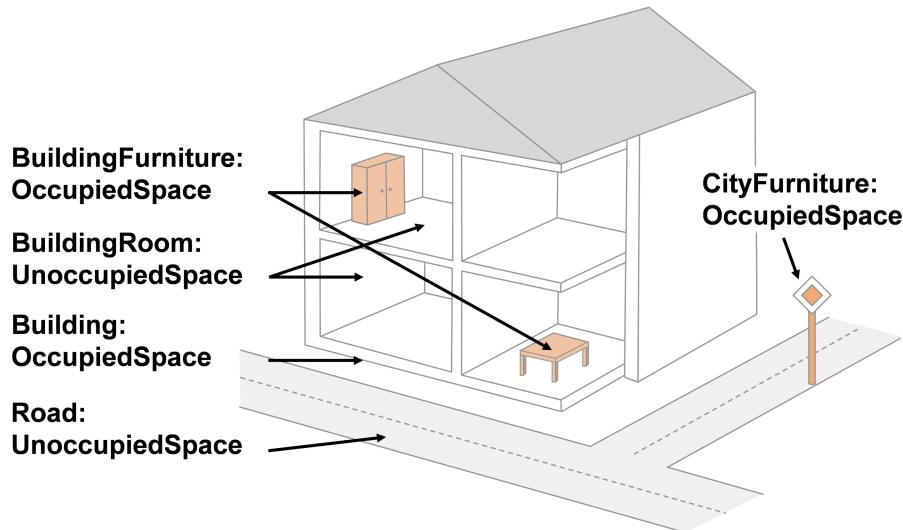


Figure 11 – Occupied and unoccupied spaces

The new space concept offers several advantages.

- In the CityGML 3.0 Conceptual Model, all geometric representations are only defined in the *Core* module. This makes (a) models of the thematic modules simpler as they no longer need to be associated directly with the geometry classes, and (b) implementation easier as all spatial concepts have only to be implemented once in the *Core* module. All thematic modules like *Building*, *Relief*, *WaterBody*, etc. inherit their geometric representations from the *Core* module.
- The space concept supports the expression of explicit topological, geometrical, and thematic relations between spaces and spaces, spaces and space boundaries, and space boundaries and space boundaries. Thus, implementing the checking of geometric-topological consistency will become easier. That is because most checks can be expressed and performed on the CityGML *Core* module and then automatically applied to all thematic modules
- For the analysis of navigable spaces (e.g., to generate IndoorGML data from CityGML) algorithms can be defined on the level of the *Core* module. These algorithms will then work with all CityGML feature classes and also ADEs as they are derived from the *Core*. The same is true for other applications of 3D city models listed in [Biljecki et al. 2015] such as visibility analyses including shadow casting or solar irradiation analyses.
- Practitioners and developers do not see much of the space concept. That is because the space and space boundary classes are just abstract classes. Only elements representing objects from concrete subclasses such as *Building*, *BuildingRoom*, or *TrafficSpace* will appear in CityGML data sets.

6.4.2. Modeling City Objects by the Composition of Spaces

Semantic objects in CityGML are often composed of parts, i.e., they form multi-level aggregation hierarchies. This also holds for semantic objects representing occupied and unoccupied spaces. In general, two types of compositions can be distinguished.

1. **Spatial partitioning:** Semantic objects of either the space type OccupiedSpace or UnoccupiedSpace are subdivided into different parts that are of the same space type as the parent object. Examples are Buildings that can be subdivided into BuildingParts, or Buildings that are partitioned into ConstructiveElements. Buildings as well as BuildingParts and constructiveElements represent OccupiedSpaces. Similarly, Roads can be subdivided into TrafficSpaces and AuxiliaryTrafficSpaces, all objects being UnoccupiedSpaces.
2. **Nesting of alternating space types:** Semantic objects of one space type contain objects that are of the opposite space type as the parent object. Examples are Buildings (OccupiedSpace) that contain BuildingRooms (UnoccupiedSpace), BuildingRooms (UnoccupiedSpace) that contain Furniture (OccupiedSpace), and Roads (UnoccupiedSpace) that contain CityFurniture (OccupiedSpace). The categorization of a semantic object into occupied or unoccupied takes place at the level of the object in relation to the parent object. A building is part of a city model. Thus, in the first place the building occupies urban space within a city. As long as the interior of the building is not modeled in detail, the space covered by the building needs to be considered as occupied and only viewable from the outside. To make the building accessible inside, voids need to be added to the building in the form of building rooms. The rooms add free space to the building interior. In other words, the OccupiedSpace now contains some UnoccupiedSpace. The free space inside the building can, in turn, contain objects that occupy space again, such as furniture or installations. In contrast, roads also occupy urban space in the city. However, this space is initially unoccupied as it is accessible by cars, pedestrian, or cyclists. Adding traffic signs or other city furniture objects to the free space results in specific sections of the road becoming occupied by these objects. Thus, one can also say that occupied spaces are mostly filled with matter; whereas, unoccupied spaces are mostly free of matter and, thus, realize free spaces.

6.4.3. Rules for Surface Orientations of OccupiedSpaces and UnoccupiedSpaces

The classification of feature types into OccupiedSpace and UnoccupiedSpace also defines the semantics of the geometries attached to the respective features. For OccupiedSpaces, the attached geometries describe volumes that are (mostly) physically occupied. For UnoccupiedSpaces, the attached geometries describe (or bound) volumes that are (mostly) physically unoccupied. This also has an impact on the required orientation of the surface normal (at point P this is a vector perpendicular to the tangent plane of the surface at P) for attached thematic surfaces. For OccupiedSpaces, the normal vectors of thematic surfaces must point

in the same direction as the surfaces of the outer shell of the volume. For UnoccupiedSpaces, the normal vectors of thematic surfaces must point in the opposite direction as the surfaces of the outer shell of the volume. This means that from the perspective of an observer of a city scene, the surface normals must always be directed towards the observer. In the case of OccupiedSpaces (e.g., Buildings, Furniture), the observer must be located outside the OccupiedSpace for the surface normals being directed towards the observer; whereas in the case of UnoccupiedSpaces (e.g., Rooms, Roads), the observer is typically inside the UnoccupiedSpace.

6.4.4. Levels of Detail (LOD)

The CityGML Conceptual Model differentiates four consecutive Levels of Detail (LOD 0-3), where objects become more detailed with increasing LOD with respect to their geometry. CityGML datasets can – but do not have to – contain multiple geometries for each object in different LODs simultaneously. The LOD concept facilitates multi-scale modeling; i.e., having varying degrees of spatial abstractions that are appropriate for different applications or visualizations.

The classification of real-world objects into spaces and space boundaries is solely based on the semantics of these objects and not on their used geometry type, as the CityGML 3.0 CM allows various geometrical representations for objects. A building, for instance, can be spatially represented by a 3D solid (e.g., in LOD1), but at the same time, the real-world geometry can also be abstracted by a single point, footprint or roof print (LOD0), or by a 3D mesh (LOD3). The outer shell of the building may also be semantically decomposed into wall, roof, and ground surfaces. Figure 12 shows different representations of the same real-world building object in different geometric LODs (and appearances).

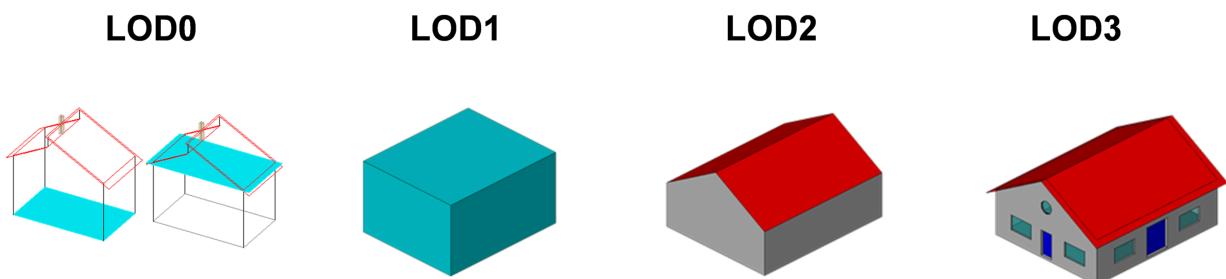


Figure 12 – Representation of the same real-world building in the Levels of Detail 0-3.

The biggest changes between CityGML 3.0 and earlier versions are the following.

1. LOD4 was dropped, because now all feature types can have outdoor and indoor elements in LODs 0-3 (for those city objects where it makes sense like buildings, tunnels, or bridges). This means that the outside shell such as of a building, could be spatially represented in LOD2 and the indoor elements like rooms, doors, hallways, stairs etc. in LOD1. CityGML can now be used to represent building floor plans, which are LOD0 representations of building interiors (cf. Konde et al. 2018). It is even possible to model the outside shell of a building in LOD1, while representing the interior structure in LOD2 or 3. Figure 13 shows different

indoor/outdoor representations of a building. Details on the changes to the CityGML LOD concept are provided in [Löwner et al. 2016].

2. Levels of Detail are no longer associated with the degree of semantic decomposition of city objects and refer to the spatial representations only. This means that, for example, buildings can have thematic surfaces (like WallSurface, GroundSurface) also in LODs 0 and 1 and windows and doors can be represented in all LODs 0-3. In CityGML 2.0 or earlier thematic surfaces were only allowed starting from LOD2, openings like doors and windows starting from LOD3, and interior rooms and furniture only in LOD4.
3. In the CityGML 3.0 Conceptual Model the geometry representations were moved from the thematic modules to the *Core* module and are now associated with the semantic concepts of *Spaces* and *Space Boundaries*. This led to a significant simplification of the models of the thematic modules. Since all feature types in the thematic modules are defined as subclasses of the space and space boundary classes, they automatically inherit the geometry classes and, thus, no longer require direct associations with them. This also led to a harmonized LOD representation over all CityGML feature types.
4. If new feature types are defined in Application Domain Extensions (ADEs) based on the abstract Space and Space Boundary classes from the Core module, they automatically inherit the spatial representations and the LOD concept.

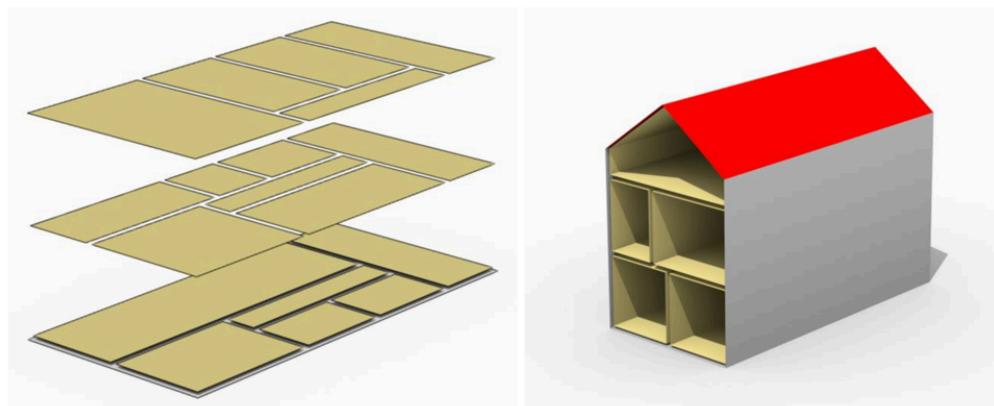


Figure 13 – Floor plan representation (LOD0) of a building (left), combined LOD2 indoor and outdoor representation (right). Image adopted from Löwner et al. 2016.

Spaces and all its subclasses like *Building*, *Room*, and *TrafficSpace* can now be spatially represented by single points in LOD0, multi-surfaces in LOD0/2/3, solids in LOD1/2/3, and multi-curves in LOD2/3. *Space Boundaries* and all its subclasses such as *WallSurface*, *LandUse*, or *Relief* can now be represented by multi-surfaces in LOD0/2/3 and as multi-curves in LOD2/3. See Clause 7.2.5 for further details on the different Levels of Detail.

6.4.5. Closure Surfaces

Objects, which are not spatially represented 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 specific type of space boundary called a ClosureSurface. These are virtual surfaces. They are used when a closed surface is needed to compute volumes or perform similar 3D operations. Since they do not actually exist, they are neglected when they are not needed or not appropriate. For example, ClosureSurfaces would not be used in visualizations.

The concept of ClosureSurface can also be employed to model the entrances of subsurface objects. Those objects like tunnels or pedestrian underpasses have to be modeled as closed solids in order to compute their volume. An example would be for use in flood simulations. The entrances to subsurface objects also have to be sealed to avoid holes in the digital terrain model (see Figure 14). However, in close-range visualizations the entrance should be treated as open. Thus, closure surfaces are an adequate way to model those entrances.

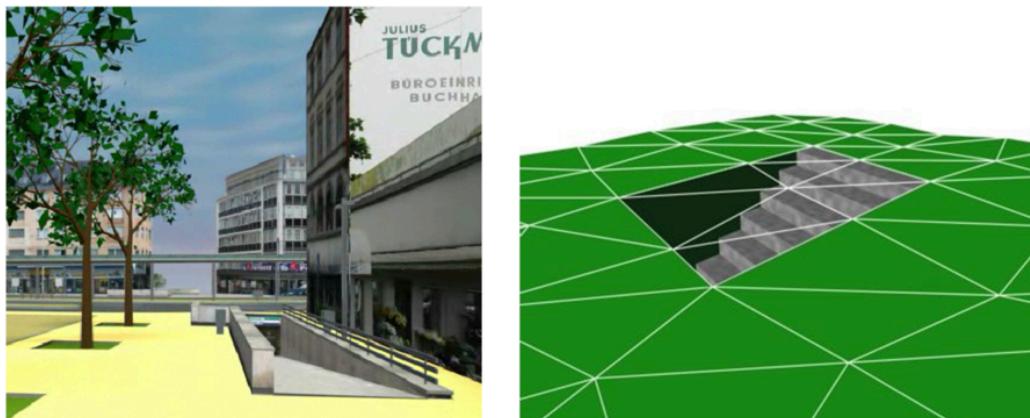


Figure 14 – Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual ClosureSurface feature, which is both part of the DTM and the subsurface object (right) (graphic: IGG Uni Bonn).

6.4.6. Terrain Intersection Curves

An important issue in city modeling 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 when terrains and 3D objects in different LODs are combined, when the terrain and 3D models are updated independently from each other, or when they come from different data providers [Kolbe 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 Figure 15). TICs can be applied to all CityGML feature types that are derived from AbstractPhysicalSpace such as buildings, bridges, tunnels, but also city furniture, vegetation, and generic city objects.

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 digital terrain model (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 LODs.

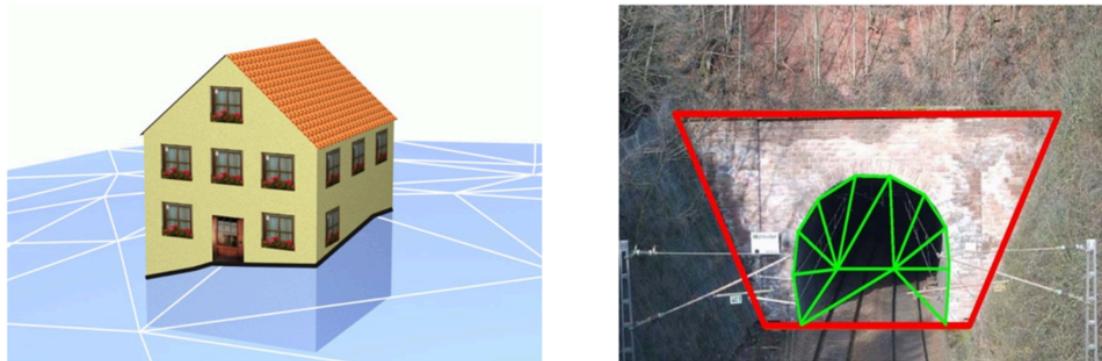


Figure 15 – `TerrainIntersectionCurve` for a building (left, black) and a tunnel object (right, red). The tunnel’s hollow space is sealed by a triangulated `ClosureSurface` (graphic: IGG Uni Bonn).

6.4.7. Coherent Semantical-Geometrical Modeling

An important design principle for CityGML is the coherent modeling of semantic objects and their spatial representations. 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, shape, 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 building is semantically decomposed into wall surfaces, roof surfaces and so forth, the polygons representing these thematic surfaces (in a specific LOD) must be part of the solid geometry representing the entire building (for the same LOD).

6.5. Appearances

In addition to semantics and geometry, information about the appearance of surfaces, i.e., observable properties of the surface, is considered an integral part of virtual 3D city and

landscape models. Appearance relates to any surface-based theme such as infrared radiation or noise pollution, not just visual properties like RGB texture images. Consequently, data provided by appearances can be used as input for both, presentation of and analysis in virtual 3D city models.

The CityGML Conceptual Model 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 the Appearance module (cf. Clause 7.3).

6.6. Modeling Dynamic Data

In general, city objects can have properties related to their geometry, topology, semantics, and appearance. All of these properties may change over time. For example, a construction event leads to the change in geometry of a building (i.e., addition of a new building floor or demolition of an existing door). The geometry of an object can be further classified according to its shape, location, and extent, which can also change over time. A moving car object involves changing only the location of the car object. However, a flood incident involves variations in the location and shape of water. There might be other properties, which change with respect to thematic data of city objects such as hourly variations in energy or gas consumption of a building or changing the building usage from residential to commercial. Some properties involve changes in appearances over a time period, such as building textures changing over years or traffic cameras recording videos of moving traffic over definite intervals. 3D city models also represent interrelationships between objects and relations may change over time as well. Hence, it is important to consider that the representation of time-varying data is required to be associated with these different properties. A detailed discussion on the requirements of city model applications regarding the support of dynamic data is given in [Chaturvedi Kolbe 2019].

The CityGML 3.0 Conceptual Model introduces two concepts to manage dynamic, time-dependent, properties of city models. The *Versioning* module manages changes that are slower in nature. Examples are (1) the history or evolution of cities such as construction or demolition of buildings, and (2) managing multiple versions of the city models.

The *Dynamizer* module manages higher-frequency or dynamic variations of object properties, including variations of (1) thematic attributes such as changes of physical quantities (energy demands, temperature, solar irradiation levels), (2) spatial properties such as change of a feature's geometry, with respect to shape and location (moving objects), and (3) real-time sensor observations. The *Dynamizer* module allows establishing explicit links from city objects to sensors and sensor data services.

6.6.1. Versioning and Histories

As described in Clause 6.2.1, the bitemporal timestamps of all CityGML feature types allow representing the evolution of the real city and its model over time. The new *Versioning* module extends this concept by the possibility of representing multiple, concurrent versions of the city model. For that purpose, the module defines two new feature types: 1) *Version*, which can be

used to explicitly define named states of the 3D city model and denote all the specific versions of objects belonging to such states. 2) *VersionTransition*, which allows to explicitly link different versions of the 3D city model by describing the reason of change and the modifications applied. Details on the versioning concept are given in [Chaturvedi et al. 2015].

This approach not only facilitates the explicit representation of different city model versions, but also allows distinguishing and referring to different versions of city objects in an interoperable exchange format. All object versions could be stored and exchanged within a single dataset. Software systems could use such a dataset to visualize and work with the different versions simultaneously. The conceptual model also takes into account the management of multiple histories or multiple interpretations of the past of a city, which is required when looking at historical city developments and for archaeological applications. In addition, the Versioning module supports collaborative work. All functionality to represent a tree of workspaces as version control systems like *git* or *SVN* is provided. The Versioning module handles versions and version transitions as feature types, which allows the version management to be completely handled using the standard OGC Web Feature Service [Vretanos 2010]. No extension of the OGC Web Feature Service standard is required to manage the versioning of city models.

6.6.2. Dynamizers: Using Time-Series Data for Object Attributes

The new Dynamizer module improves the usability of CityGML for different kinds of simulations as well as to facilitate the integration of devices from the *Internet of Things (IoT)* (Annex B.1.2.29) like sensors with 3D city models. Both, simulations and sensors provide dynamic variations of some measured or simulated properties such as the electricity consumption of a building or the traffic density within a road segment. The variations of the value are typically represented using time-series data. The data sources of the time-series data could be either sensor observations (e.g., from a smart meter), pre-recorded load profiles (e.g., from an energy company), or the results of some simulation run.

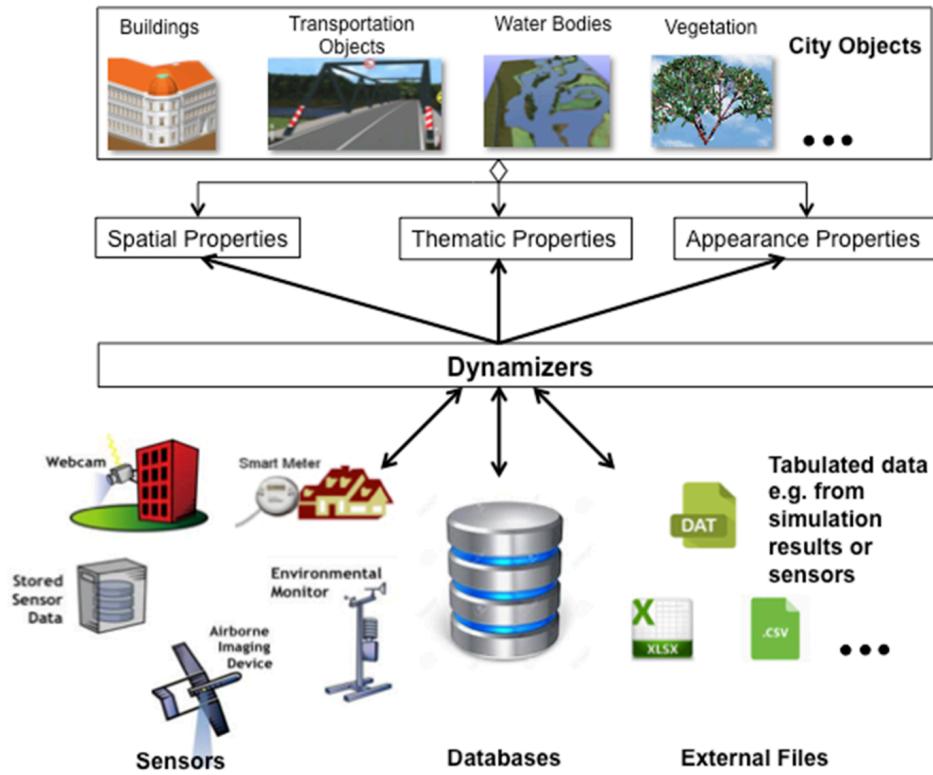


Figure 16 – Dynamizers link timeseries data coming from different sources to specific properties of individual city objects.

As shown in Figure 16, Dynamizers serve three main purposes:

1. Dynamizer is a data structure to represent dynamic values in different and generic ways. Such dynamic values may be given by (1) tabulation of time/value pairs using its *AtomicTimeseries* class, (2) patterns of time/value pairs based on statistical rules using its *CompositeTimeseries* class, and (3) retrieving observations directly from external sensor/IoT (Annex B.1.2.29) services using its *SensorConnection* class. The values can be obtained from sensor services such as the OGC Sensor Observation Service (OGC 12-006) or OGC SensorThings API (OGC 15-078r6), simulation specific databases, and also external files such as CSV or Excel sheets.
2. Dynamizer delivers a method to enhance static city models by adding dynamic property values. A Dynamizer references a specific property (e.g., spatial, thematic or appearance properties) of a specific object within a 3D city model providing dynamic values overriding the static value of the referenced object attribute.
3. Dynamizer objects establish explicit links between sensor/observation data and the respective properties of city model objects that are measured by them. By making such explicit links with city object properties, the semantics of sensor data become implicitly defined by the city model.

Dynamizers are used to inject dynamic variations of city object properties into an otherwise static representation. The advantage in following such an approach is that it allows only selected properties of city models to be made dynamic. If an application does not support dynamic data, the application simply does not allow/include these special types of features.

Dynamizers have already been implemented as an Application Domain Extension (ADE) for CityGML 2.0 and were employed in the OGC Future City Pilot Phase 1. More details about Dynamizers are given in OGC 16-098.

6.7. Extending CityGML

CityGML is designed as a universal 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 modeled in CityGML. Moreover, there might be 3D objects which are not covered by the CityGML CM thematic classes. The CityGML CM provides three different concepts to support the exchange of such data:

1. Generic objects and attributes (Clause 7.7),
2. Application Domain Extensions (Clause 9), and
3. Code lists (Clause 4.1.5).

The concept of generic objects and attributes allows application-specific concepts to be represented in CityGML at runtime. This means that any city object may be augmented by additional attributes and relations, whose names, data types, and values can be provided by a running application without requiring extensions to the CityGML conceptual schema and the respective encodings. Similarly, features not represented by the predefined thematic classes of the CityGML conceptual model may be modeled and exchanged using generic objects. The generic extensions of CityGML are provided by the *Generics* module (cf. Clause 7.7).

Application Domain Extensions (ADE) specify additions to the CityGML conceptual model. Such additions comprise the introduction of new properties to existing CityGML feature types such as the energy demand of a building or the definition of additional feature types. The difference between ADEs and generic objects and attributes is, that an ADE has to be defined in an extra conceptual schema (provided in UML) with its own namespace. Encodings have to be extended accordingly. The advantage of this approach is that the extension is formally specified. Extended CityGML datasets can be validated against the CityGML CM and the respective ADE schema. ADEs can be defined (and even standardized) by information communities which are interested in specific application fields. More than one ADE can be used simultaneously in the same dataset. Examples for popular ADEs are the Utility Network ADE [Becker et al. 2011; Kutzner et al. 2018] and the Energy ADE [Nouvel et al. 2015; Aguiaro et al. 2018]. A comprehensive overview of CityGML ADEs is given in [Biljecki et al. 2018]. Further details on ADEs are given in Clause 9.

CityGML can also be extended with regard to the allowed values specified in code lists. Many attributes of CityGML types use a code list as a data type such as, for instance, the attributes *class*, *usage*, and *function* of city objects. A code list defines a value domain including a code for each permissible value. In contrast to fixed enumerations, modifications and extensions to the value domain become possible with code lists. The values for all code lists in CityGML have to be defined externally. This could, for example, be by adopting classifications from global, national, or industrial standards.

Additional information about the extension features of CityGML can be found in the [CityGML 3.0 Users Guide](#).

7

CITYGML UML MODEL

The CityGML UML model is the normative definition of the CityGML Conceptual Model. The tables and figures in this section were software generated from the UML model. As such, this section provides a normative representation of the CityGML Conceptual Model.

An alternate representation can be found in the Data Dictionary in Clause 8.

7.1. Structural overview of requirements classes

The Requirements Classes for this Standard are structured as UML Packages as illustrated in Figure 17. Each Requirements Class is specified in detail in their respective subsections. These subsections include a UML diagram, data dictionary, and the applicable requirements.

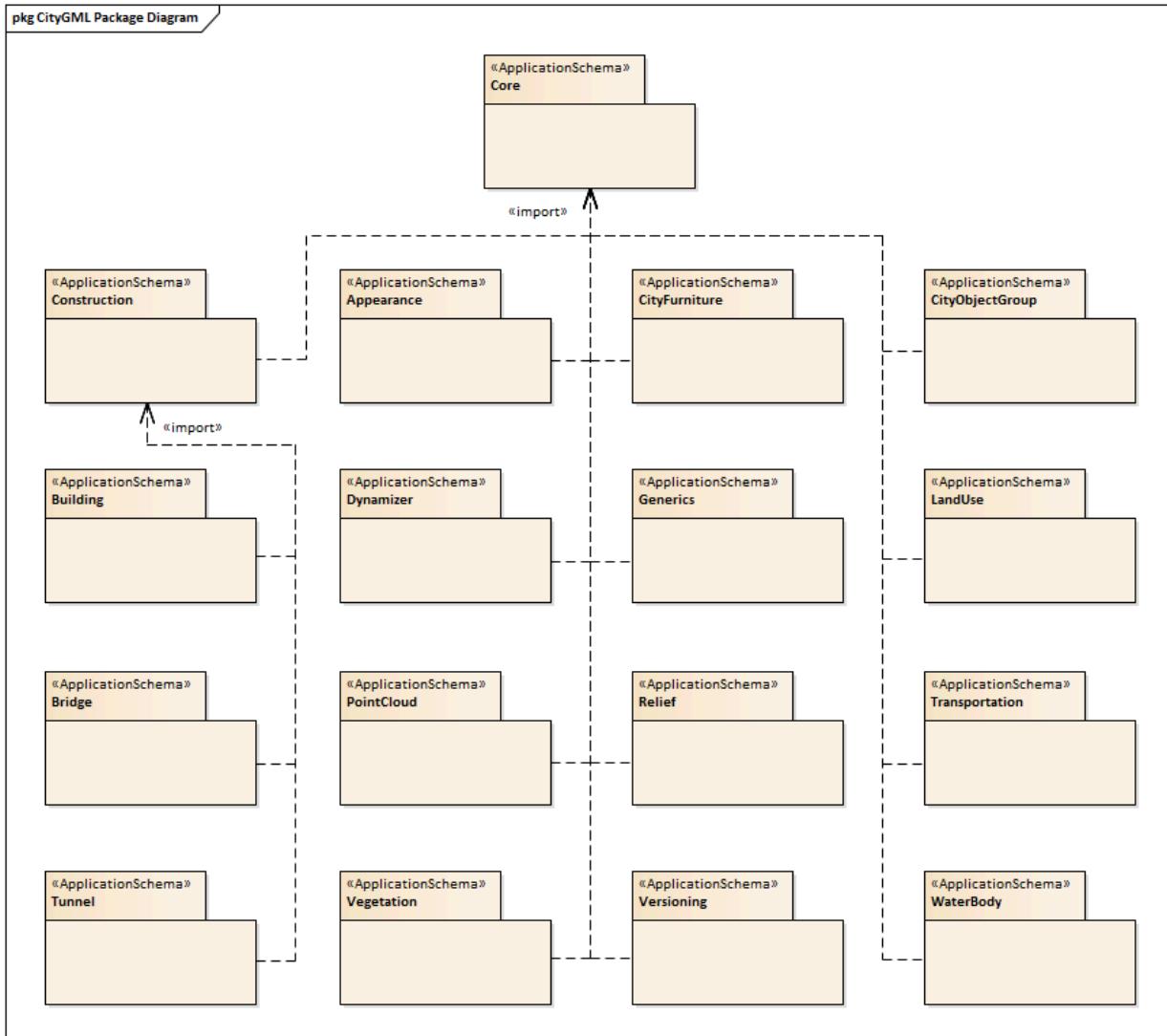


Figure 17 – CityGML UML Packages

A detailed discussion of the structuring of the UML model can be found in the [OGC CityGML 3.0 Users Guide](#).

7.2. Core

REQUIREMENTS CLASS 1

Target type	Implementation Specification
-------------	------------------------------

Dependency	ISO 19103:2015
------------	----------------

Dependency	ISO 19107:2003
------------	----------------

REQUIREMENTS CLASS 1

Dependency	ISO 19109:2015
Dependency	ISO 19111:2019
Dependency	ISO 19123:2005
Dependency	OASIS xAL v3.0

The CityGML Core module defines the basic concepts and components of city models. This rather large body of work is divided into seven sections. These sections build on each other from the fundamental principles specified by the relevant ISO standards up to the full CityGML model. These sections are summarized in Table 1.

Table 1 – CityGML Core Sections

The Use of ISO Standards (Clause 7.2.2)	Describes the use of the ISO 19100 series of International Standards to provide a foundation to the CityGML model.
City Models and City Objects (Clause 7.2.3)	Defines the basic building blocks of the CityGML model.
Space Concept (Clause 7.2.4)	Defines the concepts of space as used in the CityGML model.
Geometry and LOD (Clause 7.2.5)	Defines the geometry and Levels Of Detail concepts.
CityGML Core Model (Clause 7.2.6)	Presents the complete Core model.
Data types, Enumerations, and Code lists (Clause 7.2.7)	Defines the little things which make this model work.

Table 2 – Core space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractLogicalSpace	<ul style="list-style-type: none"> Core::AbstractSpaceBoundary and the subclasses: <ul style="list-style-type: none"> Core::AbstractThematicSurface, Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
AbstractOccupiedSpace	<ul style="list-style-type: none"> Core::AbstractSpaceBoundary and the subclasses: <ul style="list-style-type: none"> Core::AbstractThematicSurface, Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractPhysicalSpace	<ul style="list-style-type: none"> Core::AbstractSpaceBoundary and the subclasses: <ul style="list-style-type: none"> Core::AbstractThematicSurface, Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
AbstractSpace	<ul style="list-style-type: none"> Core::AbstractSpaceBoundary and the subclasses: <ul style="list-style-type: none"> Core::AbstractThematicSurface, Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
AbstractUnoccupiedSpace	<ul style="list-style-type: none"> Core::AbstractSpaceBoundary and the subclasses: <ul style="list-style-type: none"> Core::AbstractThematicSurface, Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

7.2.1. Requirements

The CityGML Core defines technology-agnostic concepts. These concepts are then realized in technology-specific Implementation Specifications. The following requirements govern the creation of any CityGML compliant Implementation Specification (IS).

REQUIREMENT 1

For each UML class defined or referenced in the Core Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.

REQUIREMENT 1

F

The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

While the CityGML Conceptual Model builds on ISO Standards, there are some restrictions on the use of those standards.

REQUIREMENT 2

ISO classes used in the CityGML Conceptual Model are subject to the following restrictions:

A

Classes derived from the GM_Solid class (ISO 19107) SHALL only include exterior boundaries. (The *interior* association on the GM_SolidBoundary shall not be defined)

An implementing technology may not be able to support all of the concepts defined in the CityGML Conceptual Model. Alternately, some concepts from the Conceptual Model may be inappropriate for the application domain for which the Implementation Specification was developed. In those cases, elements of the Conceptual Model may be mapped to null elements in the Implementation Specification.

PERMISSION 1

For each UML class defined or referenced in CityGML Conceptual Model:

A

An Implementation Specification MAY represent that class as a null class with no attributes, associations, or definition.

B

An Implementation Specification MAY represent an association of the UML class with a null association.

C

An Implementation Specification MAY represent an attribute of the UML class with a null attribute.

D

Whenever a null element is used to represent a concept from the Conceptual Model, the Implementation Specification SHOULD document that mapping and provide an explanation for why that concept was not implemented.

Surface boundaries are constrained by the following requirement:

REQUIREMENT 3

Table 2 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Core module. An Implementation Specification SHALL only support the boundaries described in Table 2

The use of extension capabilities by Core elements is constrained by the following requirement:

REQUIREMENT 4

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.2.2. ISO Dependencies

CityGML builds on the ISO 19100 family of standards. The applicable standards are identified in the diagram in Figure 18. Data dictionaries are included for all of the ISO-defined classes explicitly referenced in the CityGML UML model. These data dictionaries are provided for the convenience of the user. The ISO standards are the normative source.

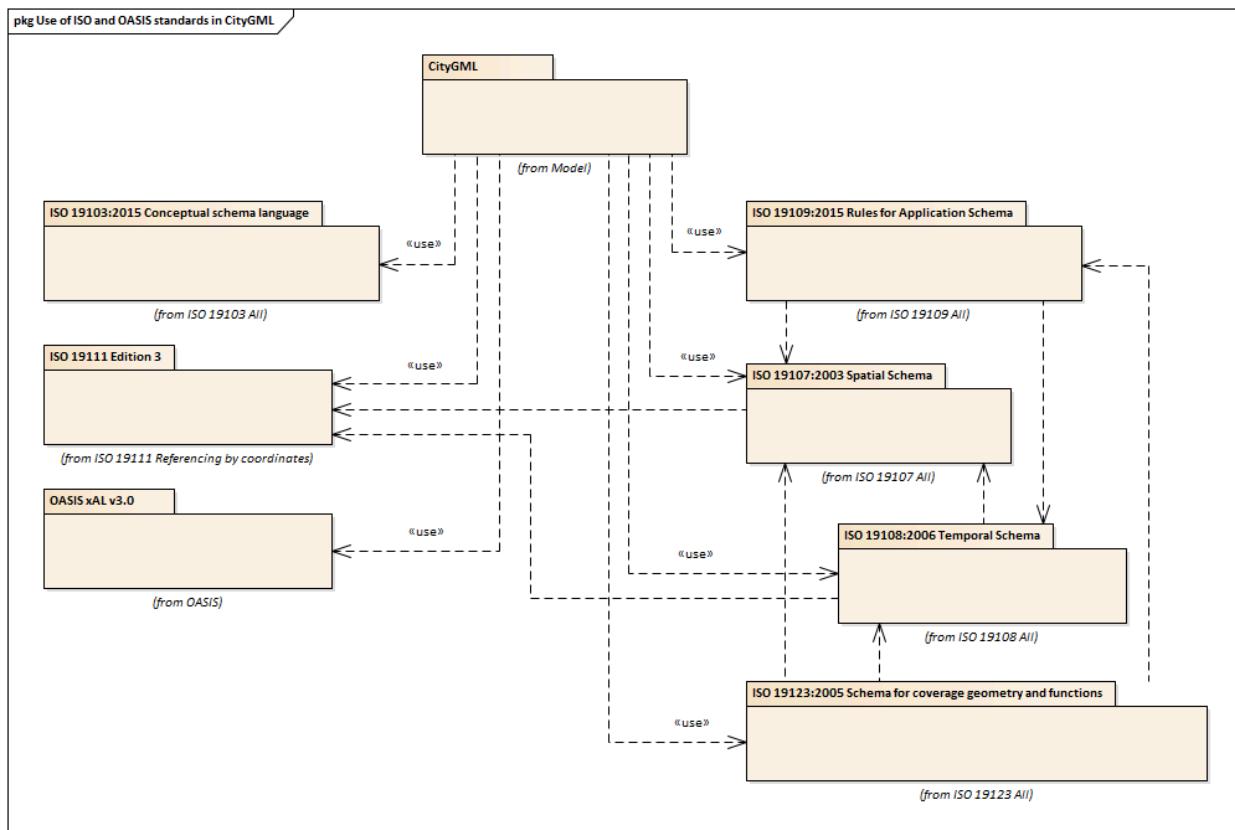


Figure 18 – Use of ISO Standards in CityGML

The ISO classes explicitly used in the CityGML UML model are introduced in Table 3. More details about these classes can be found in the Data Dictionary in Clause 8.

Table 3 – ISO Classes used in CityGML

CLASS NAME	DESCRIPTION
AnyFeature	A generalization of all feature types

CLASS NAME	DESCRIPTION
CV_DiscreteGridPointCoverage	A coverage that returns the same feature attribute values for every direct position within any object in its domain.
Direct Position	The coordinates for a position within some coordinate reference system.
GM_Object	root class of the geometric object taxonomy.
GM_MultiCurve	An aggregate class containing only instances of GM_OrientableCurve.
GM_MultiPoint	An aggregate class containing only points.
GM_MultiSurface	An aggregate class containing only instances of GM_OrientableSurface.
GM_Point	The basic data type for a geometric object consisting of one and only one point.
GM_Solid	The basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.
GM_Surface	The basis for 2-dimensional geometry.
GM_Tin	A GM_TriangulatedSurface which uses the Delaunay or similar algorithm.
GM_TriangulatedSurface	A GM_PolyhedralSurface that is composed only of triangles
SC_CRS	Coordinate reference system which is usually single but may be compound.
TM_Position	A union class that consists of one of the data types listed as its attributes.

7.2.3. City Models and City Objects

City models are virtual representations of real-world cities and landscapes. A city model aggregates different types of objects, which can be city objects, appearances, different versions of the city model, transitions between different versions of the city model, and feature objects. All objects defined in the CityGML CM are features with lifespan. This allows the optional specification of the real-world and database times for the existence of each feature, as is required by the Versioning module (cf. Clause 7.13). Features that define thematic concepts related to cities and landscapes, such as building, bridge, water body, or land use, are referred to as city objects. All city objects define properties that describe the objects in more detail. These static properties can be overridden with time-varying data through Dynamizers (cf. Clause 7.6).

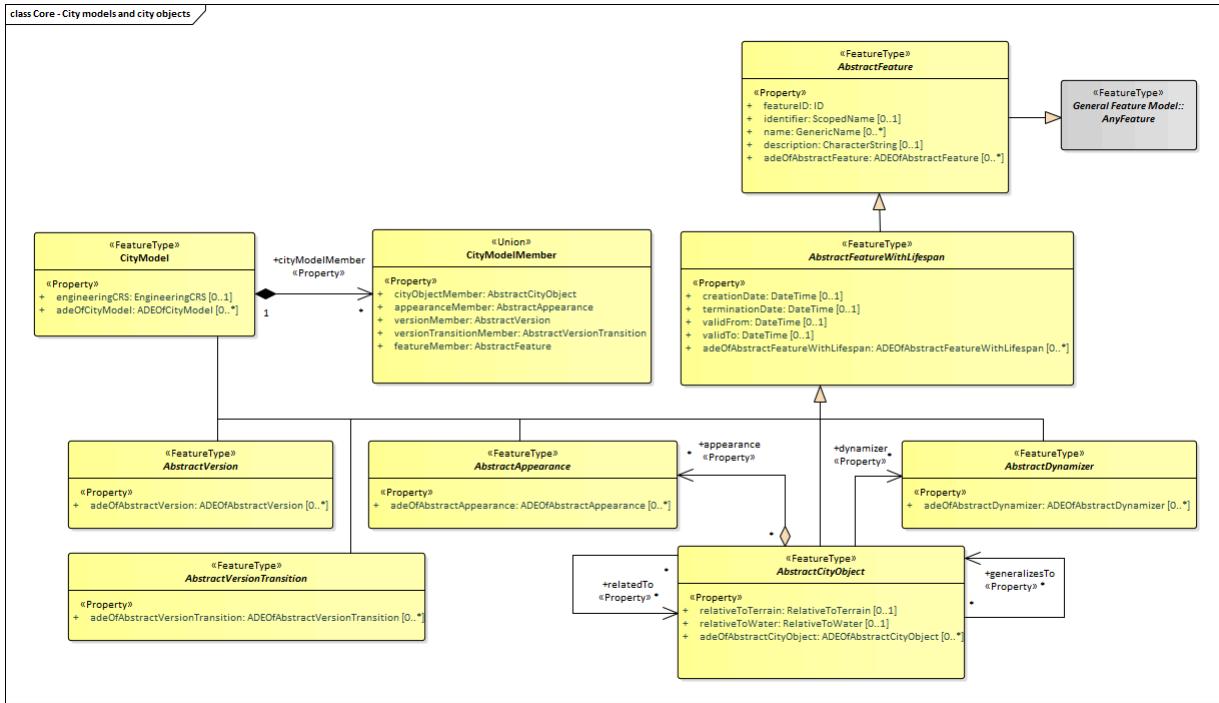


Figure 19 – UML City Models and City Objects

The City Model and City Object classes defined in the CityGML UML model are introduced in Table 4. More details about these classes can be found in the Data Dictionary in Clause 8.

Table 4 – City Model and City Object classes used in Core

CLASS	DESCRIPTION
AbstractAppearance «FeatureType»	AbstractAppearance is the abstract superclass to represent any kind of appearance objects.
AbstractCityObject «FeatureType»	AbstractCityObject is the abstract superclass of all thematic classes within the City GML Conceptual Model.
AbstractDynamizer «FeatureType»	AbstractDynamizer is the abstract superclass to represent Dynamizer objects.
AbstractFeature «Feature Type»	AbstractFeature is the abstract superclass of all feature types within the CityGML Conceptual Model.
AbstractFeatureWithLifespan «FeatureType»	AbstractFeatureWithLifespan is the base class for all CityGML features. This class allows the optional specification of the real-world and database times for the existence of each feature.
AbstractVersion «Feature Type»	AbstractVersion is the abstract superclass to represent Version objects.
AbstractVersionTransition «FeatureType»	AbstractVersionTransition is the abstract superclass to represent VersionTransition objects.

CLASS	DESCRIPTION
CityModel «FeatureType»	CityModel is the container for all objects belonging to a city model.

7.2.4. Space Concept

All city objects are differentiated into spaces and space boundaries. Spaces are entities of volumetric extent in the real world. Buildings, water bodies, trees, rooms, and traffic spaces, for instance, have a volumetric extent. Spaces can be classified into physical spaces and logical spaces. Physical spaces, in turn, can be further classified into occupied spaces and unoccupied spaces.

Space boundaries, in contrast, are entities with areal extent in the real world. Space boundaries can be differentiated into different types of thematic surfaces, such as wall surfaces and roof surfaces.

A detailed introduction to the Space concept can be found in Clause 6.4.

In particular, the classification into OccupiedSpace and UnoccupiedSpace might not always be apparent at first sight. Carports, for instance, represent an OccupiedSpace, although they are not closed and most of the space is free of matter, see Figure 20. Since a carport is a roofed, immovable structure with the purpose of providing shelter to objects (i.e., cars), carports are frequently represented as buildings in cadastres. Thus, also in CityGML, a carport should be modeled as an instance of the class Building. Since Building is transitively a subclass of OccupiedSpace, a carport is an OccupiedSpace as well. However, only in LOD1, the entire volumetric region covered by the carport would be considered as physically occupied. In LOD1, the occupied space is defined by the entire carport solid (unless a room would be defined in LOD1 that would model the unoccupied part below the roof); whereas in LOD2 and LOD3, the solids represent more realistically the really physically occupied space of the carport. In addition, for all OccupiedSpaces, the normal vectors of the thematic surfaces like the RoofSurface need to point away from the solids, i.e., consistent with the solid geometry.

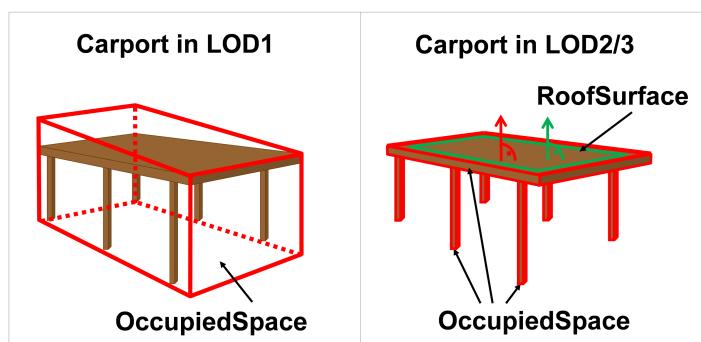


Figure 20 – Representation of a carport as OccupiedSpace in different LODs. The red boxes represent solids, the green area represents a surface. In addition, the normal vectors of the roof solid (in red) and the roof surface (in green) are shown.

In contrast, a room is a physically unoccupied space. In CityGML, a room is represented by the class BuildingRoom that is a subclass of UnoccupiedSpace. In LOD1, the entire room

solid would be considered as unoccupied space, which can contain furniture and installations, though, as is shown in Figure 21. In LOD2 and 3, the solid represents more realistically the really physically unoccupied space of the room (possibly somewhat generalized as indicated in the figure). For all UnoccupiedSpaces, the normal vectors of the bounding thematic surfaces like the InteriorWallSurface need to point inside the object, i.e., opposite to the solid geometry.

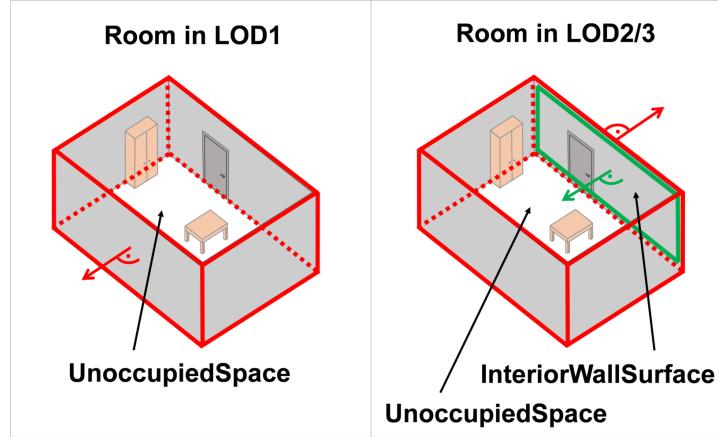


Figure 21 – Representation of a room as UnoccupiedSpace in different LODs. The red boxes represent solids, the green area represents a surface. In addition, the normal vectors of the room solid (in red) and the wall surface (in green) are shown.

The UML diagram of the Space concept classes is depicted in Figure 22.

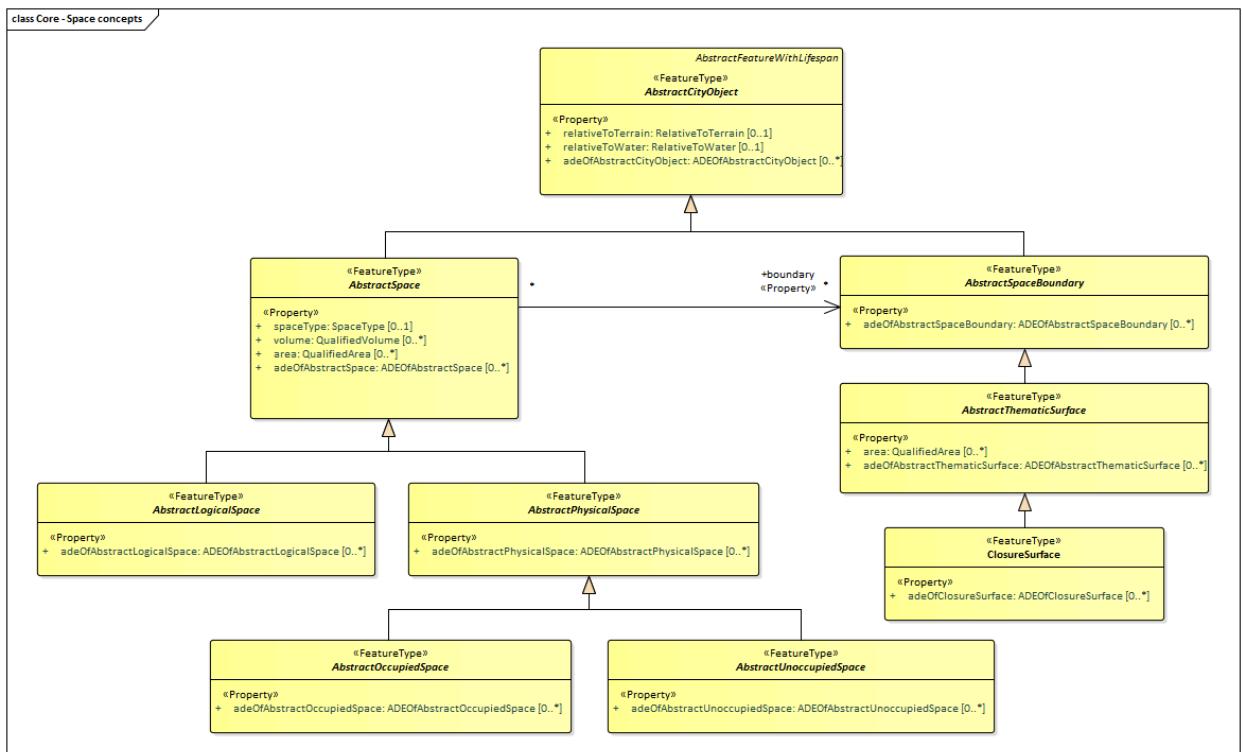


Figure 22 – UML Space Concepts

The Space Concept classes defined in the CityGML UML model are introduced in Table 5. More details about these classes can be found in the Data Dictionary in Clause 8.

Table 5 – Space Classes used in Core

CLASS	DESCRIPTION
AbstractLogicalSpace «FeatureType»	AbstractLogicalSpace is the abstract superclass for all types of logical spaces. Logical space refers to spaces that are not bounded by physical surfaces but are defined according to thematic considerations.
AbstractOccupiedSpace «FeatureType»	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
AbstractPhysicalSpace «FeatureType»	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
AbstractSpace «Feature Type»	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.
AbstractSpaceBoundary «FeatureType»	AbstractSpaceBoundary is the abstract superclass for all types of space boundaries. A space boundary is an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
AbstractThematicSurface «FeatureType»	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
AbstractUnoccupiedSpace «FeatureType»	AbstractUnoccupiedSpace is the abstract superclass for all types of physically unoccupied spaces. Unoccupied space refers to spaces that are entirely or mostly free of matter.
ClosureSurface «Feature Type»	ClosureSurface is a special type of thematic surface used to close holes in volumetric objects. Closure surfaces are virtual (non-physical) surfaces.

7.2.5. Geometry and LOD

Spaces and space boundaries can have various geometry representations depending on the Levels of Detail (LOD). Spaces can be spatially represented as single points in LOD0, multi-surfaces in LOD0/2/3, solids in LOD1/2/3, and multi-curves in LOD2/3. Space boundaries can be represented as multi-surfaces in LOD0/2/3 and as multi-curves in LOD2/3. All Levels of Detail allow for the representation of the interior of city objects.

The different Levels of Detail are defined in the following way.

- LOD 0: Volumetric real-world objects (Spaces) can be spatially represented by a single point, by a set of curves, or by a set of surfaces. Areal real-world objects (Space Boundaries) can be spatially represented in LOD0 by a set of curves or a set of surfaces. LOD0 surface representations are typically the result of a projection of the shape of a volumetric object onto a plane parallel to the ground, hence, representing a footprint (e.g., a building footprint or a floor plan of the rooms inside a building). LOD0 curve representations are either the result of a projection of the shape of a vertical surface (e.g.,

a wall surface) onto a grounding plane or the skeleton of a volumetric shape of longitudinal extent such as a road or river segment.

- LOD 1: Volumetric real-world objects (Spaces) are spatially represented by a vertical extrusion solid, i.e., a solid created from a horizontal footprint by vertical extrusion. Areal real-world objects (Space Boundaries) can be spatially represented in LOD1 by a set of horizontal or vertical surfaces.
- LOD 2: Volumetric real-world objects (Spaces) can be spatially represented by a set of curves, a set of surfaces, or a single solid geometry. Areal real-world objects (Space Boundaries) can be spatially represented in LOD2 by a set of surfaces. The shape of the real-world object is generalized in LOD2 and smaller details (e.g., bulges, dents, sills, but also structures, like balconies or dormers of buildings) are typically neglected. LOD2 curve representations are skeletons of volumetric shapes of longitudinal extent like an antenna or a chimney.
- LOD 3: Volumetric real-world objects (Spaces) can be spatially represented by a set of curves, a set of surfaces, or a single solid geometry. Areal real-world objects (Space Boundaries) can be spatially represented in LOD3 by a set of surfaces. LOD3 is the highest level of detail and respective geometries include all available shape details.

In addition, the geometry can also be represented implicitly. The shape is stored only once as a prototypical geometry, which then is re-used or referenced, wherever the corresponding feature occurs in the 3D city model.

The thematic classes, such as building, tunnel, road, land use, water body, or city furniture are defined as subclasses of the space and space boundary classes within the thematic modules. Since all city objects in the thematic modules represent subclasses of the space and space boundary classes, they automatically inherit the geometries defined in the Core module.

The UML diagram of the Geometry and LoD concept classes is depicted in Figure 23.

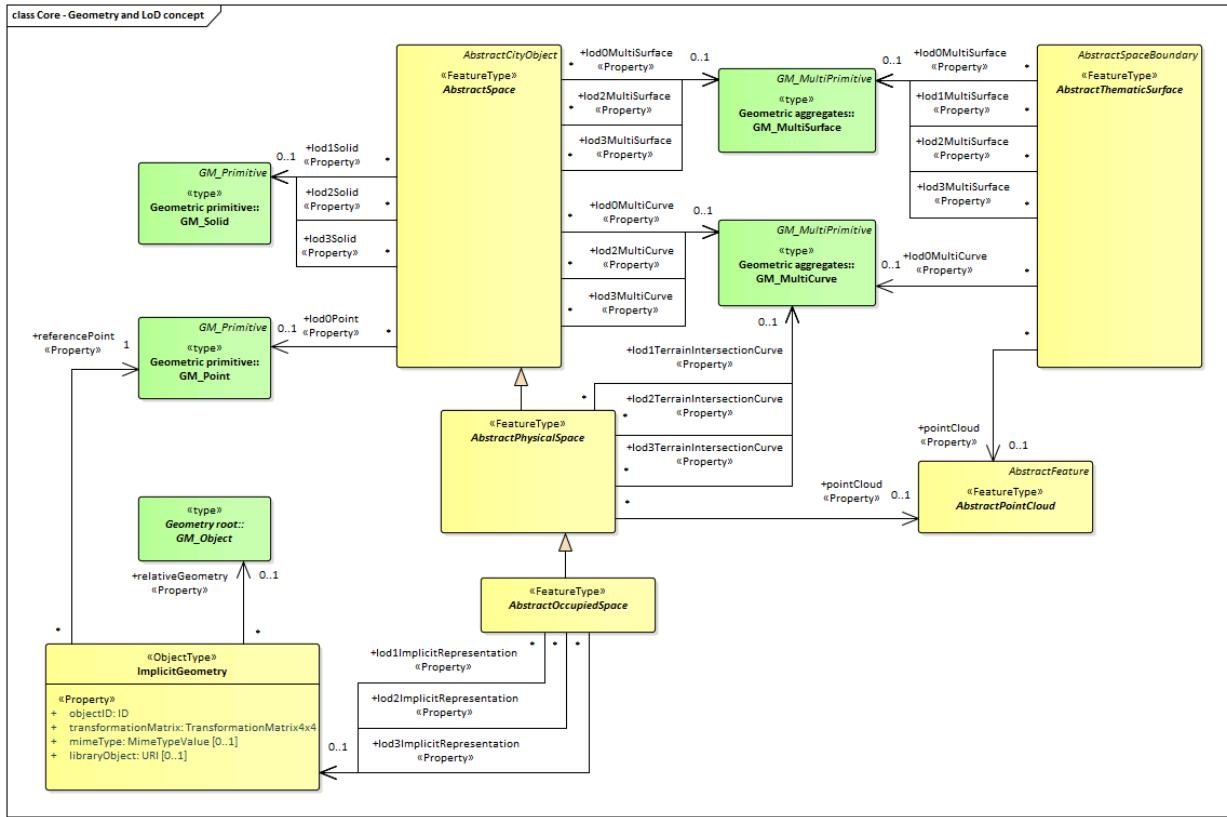


Figure 23 – UML Geometry and LOD Concepts

The Geometry and LOD Concept classes defined in the CityGML UML model are introduced in Table 6. More details about these classes can be found in the Data Dictionary in Clause 8.

Of particular note is the Implicit Geometry concept. Many of the objects encountered in a city landscape have the same geometry. How many types of street lamps can there be? An Implicit Geometry captures that geometry once, and re-uses that one geometry for all similar street lamp objects.

Table 6 – Geometry Classes used in Core

CLASS	DESCRIPTION
AbstractOccupiedSpace «FeatureType»	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
AbstractPhysicalSpace «FeatureType»	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
AbstractPointCloud «FeatureType»	AbstractPointCloud is the abstract superclass to represent PointCloud objects.
AbstractSpace «Feature Type»	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.

CLASS	DESCRIPTION
AbstractThematicSurface «FeatureType»	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
ImplicitGeometry «Object Type»	ImplicitGeometry is a geometry representation where the shape is stored only once as a prototypical geometry. Examples are a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object can be re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model.

7.2.6. CityGML Core Model

The City Model and City Object classes (Clause 7.2.3), the Space Concept classes (Clause 7.2.4), and the Geometry and LOD classes (Clause 7.2.5) define the majority of the CityGML Core module. In addition to these concepts, the Core module also specifies that city objects can have relations to other city objects and that they can have address information. All other modules defined in the CityGML model refer to the Core module.

The UML diagram of the complete Core module is depicted in Figure 24.

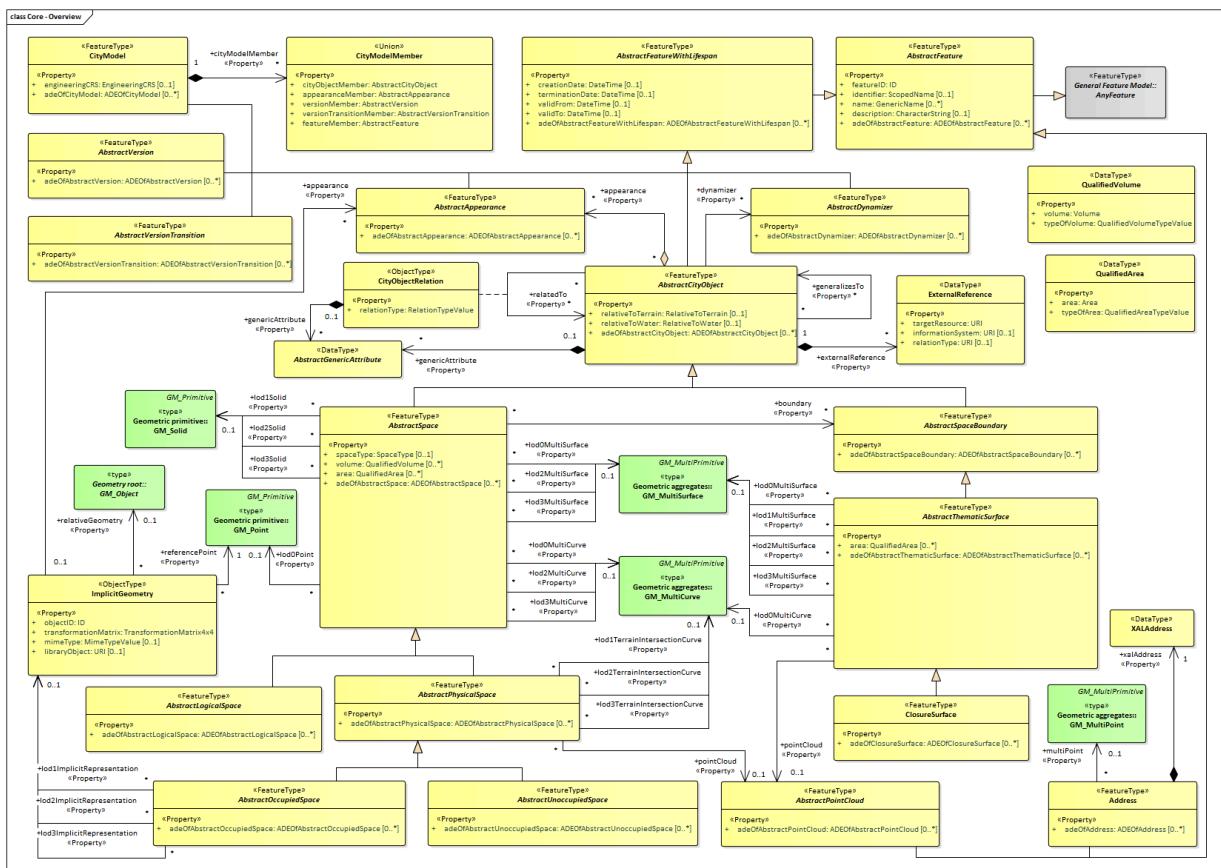


Figure 24 – UML diagram of CityGML’s core module

Table 4, Table 5 and Table 6 introduce already most of the classes of the CityGML Core module. The additional classes required to complete this section of the standard are introduced in Table 7. More details about these classes can be found in the Data Dictionary in Clause 8.

Table 7 – Additional Classes used in Core

CLASS	DESCRIPTION
Address «FeatureType»	Address represents an address of a city object.
CityObjectRelation «Object Type»	CityObjectRelation represents a specific relation from the city object in which the relation is included to another city object.

7.2.7. Data types, Enumerations, and Code lists

The ADE data types provided for in the Core module are illustrated in Figure 25.

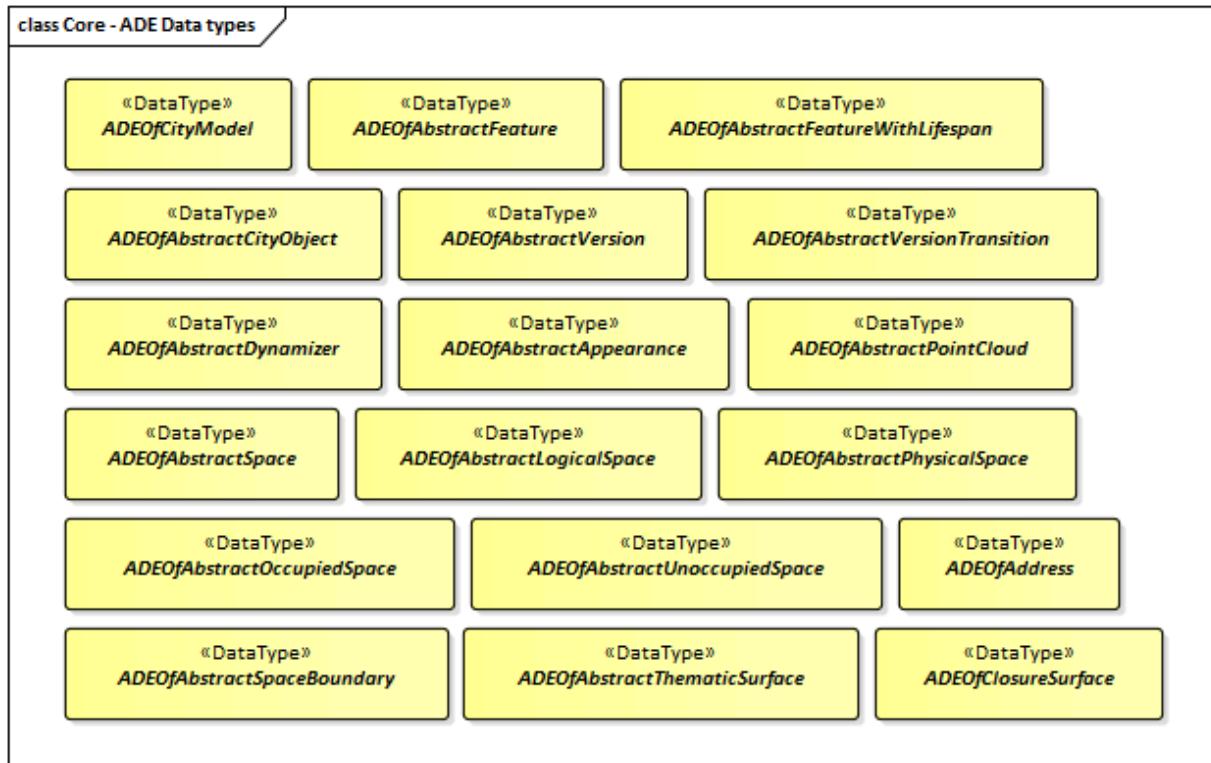


Figure 25 – ADE classes of the CityGML Core module.

The Data Types, Basic Types, Enumerations, Unions, and Code Lists provided for the Core module are illustrated in Figure 26.

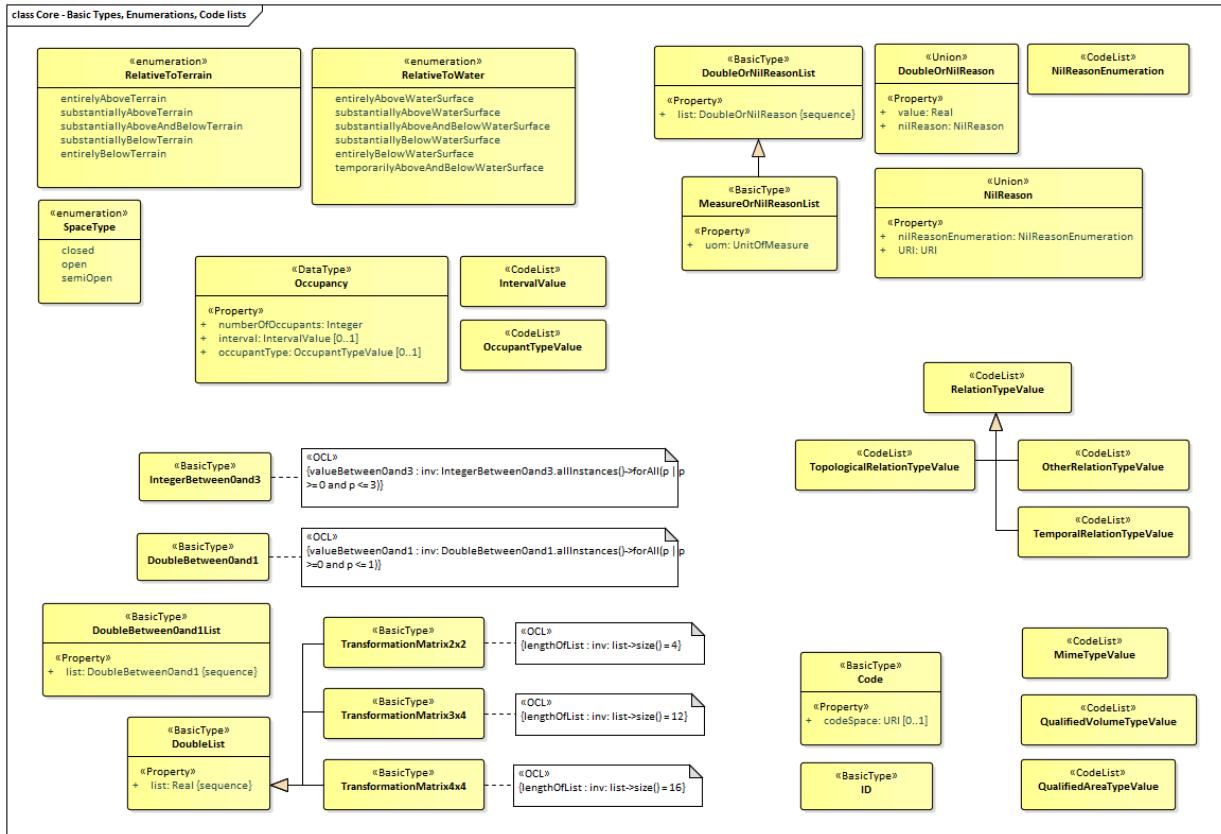


Figure 26 – Basic Types, Enumerations, and Codelists from the CityGML Core module.

These supporting constructs are defined in the following tables.

Table 8 – Data types defined in Core (ApplicationSchema)

NAME	DESCRIPTION
AbstractGenericAttribute «DataType»	AbstractGenericAttribute is the abstract superclass for all types of generic attributes.
ADEOfAbstractAppearance «DataType»	ADEOfAbstractAppearance acts as a hook to define properties within an ADE that are to be added to AbstractAppearance.
ADEOfAbstractCityObject «DataType»	ADEOfAbstractCityObject acts as a hook to define properties within an ADE that are to be added to AbstractCityObject.
ADEOfAbstractDynamizer «DataType»	ADEOfAbstractDynamizer acts as a hook to define properties within an ADE that are to be added to AbstractDynamizer.
ADEOfAbstractFeature «DataType»	ADEOfAbstractFeature acts as a hook to define properties within an ADE that are to be added to AbstractFeature.
ADEOfAbstractFeatureWithLifespan «DataType»	ADEOfAbstractFeatureWithLifespan acts as a hook to define properties within an ADE that are to be added to AbstractFeatureWithLifespan.
ADEOfAbstractLogicalSpace «DataType»	ADEOfAbstractLogicalSpace acts as a hook to define properties within an ADE that are to be added to AbstractLogicalSpace.

NAME	DESCRIPTION
ADEOfAbstractOccupiedSpace «DataType»	ADEOfAbstractOccupiedSpace acts as a hook to define properties within an ADE that are to be added to AbstractOccupiedSpace.
ADEOfAbstractPhysicalSpace «DataType»	ADEOfAbstractPhysicalSpace acts as a hook to define properties within an ADE that are to be added to AbstractPhysicalSpace.
ADEOfAbstractPointCloud «DataType»	ADEOfAbstractPointCloud acts as a hook to define properties within an ADE that are to be added to AbstractPointCloud.
ADEOfAbstractSpace «DataType»	ADEOfAbstractSpace acts as a hook to define properties within an ADE that are to be added to AbstractSpace.
ADEOfAbstractSpaceBoundary «DataType»	ADEOfAbstractSpaceBoundary acts as a hook to define properties within an ADE that are to be added to AbstractSpaceBoundary.
ADEOfAbstractThematicSurface «DataType»	ADEOfAbstractThematicSurface acts as a hook to define properties within an ADE that are to be added to AbstractThematicSurface.
ADEOfAbstractUnoccupiedSpace «DataType»	ADEOfAbstractUnoccupiedSpace acts as a hook to define properties within an ADE that are to be added to AbstractUnoccupiedSpace.
ADEOfAbstractVersion «DataType»	ADEOfAbstractVersion acts as a hook to define properties within an ADE that are to be added to AbstractVersion.
ADEOfAbstractVersionTransition «DataType»	ADEOfAbstractVersionTransition acts as a hook to define properties within an ADE that are to be added to AbstractVersionTransition.
ADEOfAddress «DataType»	ADEOfAddress acts as a hook to define properties within an ADE that are to be added to an Address.
ADEOfCityModel «DataType»	ADEOfCityModel acts as a hook to define properties within an ADE that are to be added to a CityModel.
ADEOfClosureSurface «DataType»	ADEOfClosureSurface acts as a hook to define properties within an ADE that are to be added to a ClosureSurface.
ExternalReference «DataType»	ExternalReference is a reference to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS UK MasterMap®.
Occupancy «DataType»	Occupancy is an application-dependent indication of what is contained by a feature.
QualifiedArea «DataType»	QualifiedArea is an application-dependent measure of the area of a space or of a thematic surface.
QualifiedVolume «DataType»	QualifiedVolume is an application-dependent measure of the volume of a space.
XALAddress «DataType»	XALAddress represents address details according to the OASIS xAL standard.

Table 9 – Primitive data types defined in Core (ApplicationSchema)

NAME	DESCRIPTION
Code «BasicType»	Code is a basic type for a String-based term, keyword, or name that can additionally have a code space.
DoubleBetween0and1 «BasicType»	DoubleBetween0and1 is a basic type for values, which are greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
DoubleBetween0and1List «BasicType»	DoubleBetween0and1List is a basic type that represents a list of double values greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.
DoubleList «BasicType»	DoubleList is an ordered sequence of double values.
DoubleOrNilReasonList «BasicType»	DoubleOrNilReasonList is a basic type that represents a list of double values and/or nil reasons.
ID «BasicType»	ID is a basic type that represents a unique identifier.
IntegerBetween0and3 «BasicType»	IntegerBetween0and3 is a basic type for integer values, which are greater or equal than 0 and less or equal than 3. The type is used for encoding the LOD number.
MeasureOrNilReasonList «BasicType»	MeasureOrNilReasonList is a basic type that represents a list of double values and/or nil reasons together with a unit of measurement.
TransformationMatrix2×2 «BasicType»	TransformationMatrix2×2 is a 2 by 2 matrix represented as a list of four double values in row major order.
TransformationMatrix3×4 «BasicType»	TransformationMatrix3×4 is a 3 by 4 matrix represented as a list of twelve double values in row major order.
TransformationMatrix4×4 «BasicType»	TransformationMatrix4×4 is a 4 by 4 matrix represented as a list of sixteen double values in row major order.

Table 10 – Enumerated classes defined in Core (ApplicationSchema)

NAME	DESCRIPTION
RelativeToTerrain «Enumeration»	RelativeToTerrain enumerates the spatial relations of a city object relative to terrain in a qualitative way.
RelativeToWater «Enumeration»	RelativeToWater enumerates the spatial relations of a city object relative to the water surface in a qualitative way.
SpaceType «Enumeration»	SpaceType is an enumeration that characterises a space according to its closure properties.

Table 11 – Union types defined in Core (ApplicationSchema)

NAME	DESCRIPTION
CityModelMember «Union»	CityModelMember is a union type that enumerates the different types of objects that can occur as members of a city model.
DoubleOrNilReason «Union»	DoubleOrNilReason is a union type that allows for choosing between a double value and a nil reason.
NilReason «Union»	NilReason is a union type that allows for choosing between two different types of nil reason.

Table 12 – Code list classes defined in Core (ApplicationSchema)

NAME	DESCRIPTION
IntervalValue «CodeList»	IntervalValue is a code list used to specify a time period.
MimeTypeValue «CodeList»	MimeTypeValue is a code list used to specify the MIME type of a referenced resource.
NilReasonEnumeration «CodeList»	NilReasonEnumeration is a code list that enumerates the different nil reasons.
OccupantTypeValue «CodeList»	OccupantTypeValue is a code list used to classify occupants.
OtherRelationTypeValue «CodeList»	OtherRelationTypeValue is a code list used to classify other types of city object relations.
QualifiedAreaTypeValue «CodeList»	QualifiedAreaTypeValue is a code list used to specify area types.
QualifiedVolumeTypeValue «CodeList»	QualifiedVolumeTypeValue is a code list used to specify volume types.
RelationTypeValue «CodeList»	RelationTypeValue is a code list used to classify city object relations.
TemporalRelationTypeValue «CodeList»	TemporalRelationTypeValue is a code list used to classify temporal city object relations.
TopologicalRelationTypeValue «CodeList»	TopologicalRelationTypeValue is a code list used to classify topological city object relations.

7.2.8. Additional Information

Additional information about the Core Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.3. Appearance

REQUIREMENTS CLASS 2

Target type	Implementation Specification
Dependency	/req/req-class-core

The Appearance module provides the representation of surface data such as observable properties for surface geometry objects in the form of textures and material.

Appearances are not limited to visual data but represent arbitrary categories called themes such as infrared radiation, noise pollution, or earthquake-induced structural stress. 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).

Surface data that is constant across a surface is modeled as material based on the material definitions from the X3D and COLLADA standards. Surface data that depends on the exact location within the surface is modeled as a texture. This can either be a parameterized texture (a texture that uses texture coordinates) or a transformation matrix for parameterization, or a georeferenced texture (a texture that uses a planimetric projection).

Each surface geometry object can have both, a material and a texture per theme and side. This allows for providing a constant approximation and a complex measurement of a surface's property simultaneously.

The UML diagram of the Appearance module is illustrated in Figure 27. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

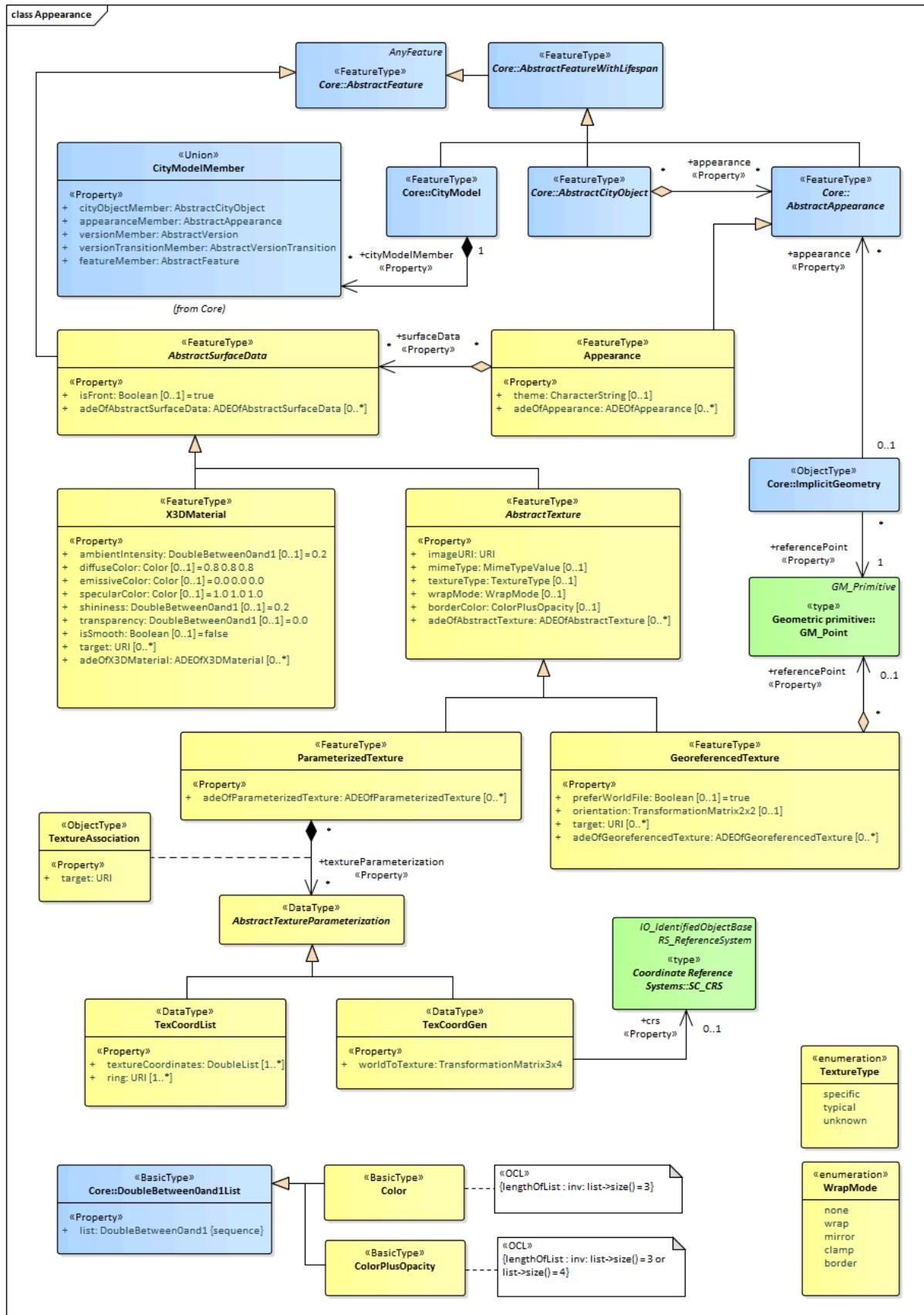


Figure 27 – UML diagram of CityGML’s Appearance model.

The ADE data types provided for the Appearance module are illustrated in Figure 28.

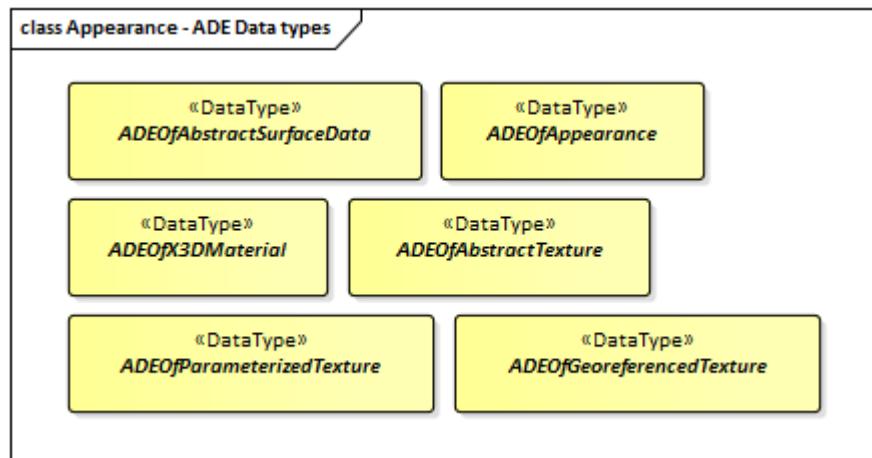


Figure 28 – ADE classes of the CityGML Appearance Module.

7.3.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Appearance Module as an Implementation Specification.

REQUIREMENT 5

For each UML class defined or referenced in the Appearance Package:

- A** The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
- B** The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
- C** The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
- D** The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
- E** The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
- F** The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Appearance elements is constrained by the following requirement:

REQUIREMENT 6

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.3.2. Class definitions

Table 13 – Classes defined in Appearance (ApplicationSchema)

NAME	DESCRIPTION
AbstractSurfaceData «FeatureType»	AbstractSurfaceData is the abstract superclass for different kinds of textures and material.
AbstractTexture «Feature Type»	AbstractTexture is the abstract superclass to represent the common attributes of the classes ParameterizedTexture and GeoreferencedTexture.
Appearance «FeatureType»	An Appearance is a collection of surface data, i.e. observable properties for surface geometry objects in the form of textures and material.
GeoreferencedTexture «FeatureType»	A GeoreferencedTexture is a texture that uses a planimetric projection. It contains an implicit parameterization that is either stored within the image file, an accompanying world file or specified using the orientation and referencePoint elements.
ParameterizedTexture «FeatureType»	A ParameterizedTexture is a texture that uses texture coordinates or a transformation matrix for parameterization.
TextureAssociation «Object Type»	TextureAssociation denotes the relation of a texture to a surface geometry object.
X3DMaterial «FeatureType»	X3DMaterial defines properties for surface geometry objects based on the material definitions from the X3D and COLLADA standards.

Table 14 – Data types defined in Appearance (ApplicationSchema)

NAME	DESCRIPTION
AbstractTexture Parameterization «Data Type»	AbstractTextureParameterization is the abstract superclass for different kinds of texture parameterizations.
ADEOfAbstractSurfaceData «DataType»	ADEOfAbstractSurfaceData acts as a hook to define properties within an ADE that are to be added to AbstractSurfaceData.
ADEOfAbstractTexture «DataType»	ADEOfAbstractTexture acts as a hook to define properties within an ADE that are to be added to AbstractTexture.
ADEOfAppearance «Data Type»	ADEOfAppearance acts as a hook to define properties within an ADE that are to be added to an Appearance.

NAME	DESCRIPTION
ADEOfGeoreferencedTexture «DataType»	ADEOfGeoreferencedTexture acts as a hook to define properties within an ADE that are to be added to a GeoreferencedTexture.
ADEOfParameterizedTexture «DataType»	ADEOfParameterizedTexture acts as a hook to define properties within an ADE that are to be added to a ParameterizedTexture.
ADEOfX3DMaterial «DataType»	ADEOfX3DMaterial acts as a hook to define properties within an ADE that are to be added to an X3DMaterial.
TexCoordGen «DataType»	TexCoordGen defines texture parameterization using a transformation matrix.
TexCoordList «DataType»	TexCoordList defines texture parameterization using texture coordinates.

Table 15 – Primitive data types defined in Appearance (ApplicationSchema)

NAME	DESCRIPTION
Color «BasicType»	Color is a list of three double values between 0 and 1 defining an RGB color value.
ColorPlusOpacity «BasicType»	Color is a list of four double values between 0 and 1 defining an RGBA color value. Opacity value of 0 means transparent.

Table 16 – Enumerated classes defined in Appearance (ApplicationSchema)

NAME	DESCRIPTION
TextureType «Enumeration»	TextureType enumerates the different texture types.
WrapMode «Enumeration»	WrapMode enumerates the different fill modes for textures.

7.3.3. Additional Information

Additional information about the Appearance Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.4. CityFurniture

REQUIREMENTS CLASS 3

Target type Implementation Specification

The CityFurniture module provides the representation of objects or pieces of equipment that are installed in the outdoor environment for various purposes, such as decoration, explanation, or control. City furniture objects are relatively small, immovable objects and usually are of prototypical or geotypical form. Examples include road signs, traffic signals, bicycle racks, street lamps, fountains, flower buckets, advertising columns, and benches.

City furniture is represented in the UML model by the top-level feature type *CityFurniture*, which is also the only class of the CityFurniture module.

The UML diagram of the CityFurniture module is depicted in Figure 29. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

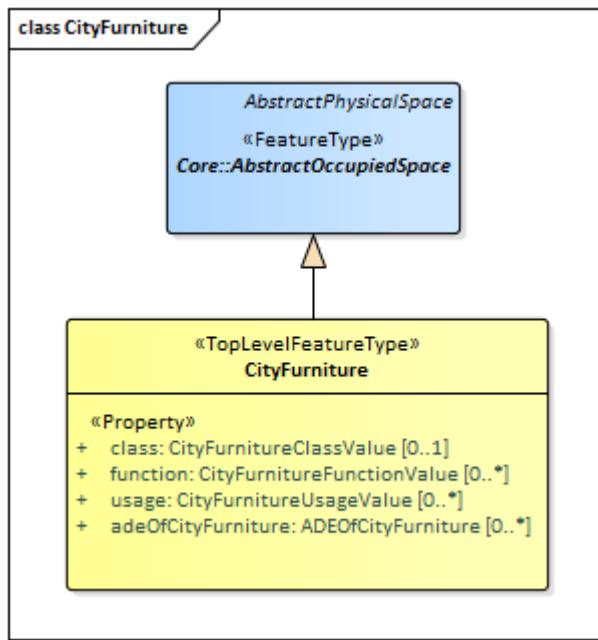


Figure 29 – UML diagram of CityGML’s City Furniture model.

The ADE data types and Code Lists provided for the CityFurniture module are illustrated in Figure 30.

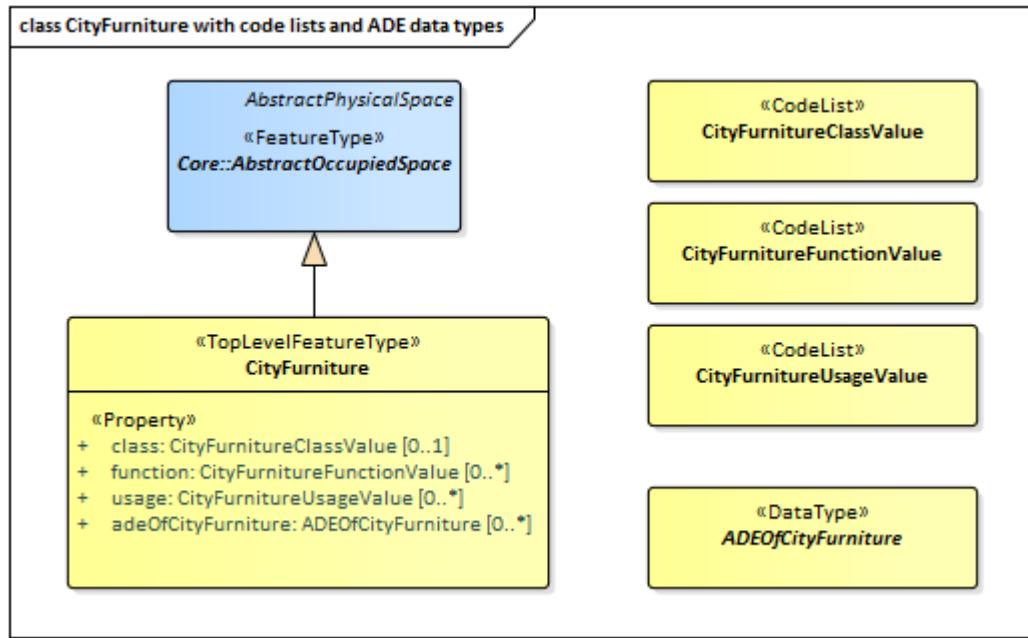


Figure 30 – ADE classes and Code Lists of the CityGML CityFurniture module.

Table 17 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the CityFurniture module:

Table 17 – CityFurniture space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
CityFurniture	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface Possible classes from ADEs

7.4.1. Requirements

The following requirement defines the rules governing implementation of the CityGML City Furniture Module as an Implementation Specification.

REQUIREMENT 7

For each UML class defined or referenced in the CityFurniture Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.

REQUIREMENT 7

C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 8

Table 17 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the CityFurniture module. An Implementation Specification SHALL only support the boundaries described in Table 17

The use of extension capabilities by City Furniture elements is constrained by the following requirement:

REQUIREMENT 9

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.4.2. Class definitions

Table 18 – Classes defined in CityFurniture (ApplicationSchema)

NAME	DESCRIPTION
CityFurniture «TopLevel FeatureType»	CityFurniture is an object or piece of equipment installed in the outdoor environment for various purposes. Examples include street signs, traffic signals, street lamps, benches, fountains.

Table 19 – Data types defined in CityFurniture (ApplicationSchema)

NAME	DESCRIPTION
ADEOfCityFurniture «Data Type»	ADEOfCityFurniture acts as a hook to define properties within an ADE that are to be added to a CityFurniture.

Table 20 – Code list classes defined in CityFurniture (ApplicationSchema)

NAME	DESCRIPTION
CityFurnitureClassValue «CodeList»	CityFurnitureClassValue is a code list used to further classify a CityFurniture.
CityFurnitureFunctionValue «CodeList»	CityFurnitureFunctionValue is a code list that enumerates the different purposes of a CityFurniture.
CityFurnitureUsageValue «CodeList»	CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

7.4.3. Additional Information

Additional information about the CityFurniture Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.5. CityObjectGroup

REQUIREMENTS CLASS 4

Target type	Implementation Specification
Dependency	/req/req-class-core

The CityObjectGroup module provides the application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group. City object groups are represented in the UML model by the top-level feature type *CityObjectGroup*, which is the main class of the CityObjectGroup module.

City object groups can be linked to other city objects, the so-called parent objects, which allows for modeling a generic hierarchical grouping concept. In addition, as city object groups represent city objects themselves, a group may become a member of another group realizing recursive aggregation in this way.

The UML diagram of the CityObjectGroup module is depicted in Figure 31. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

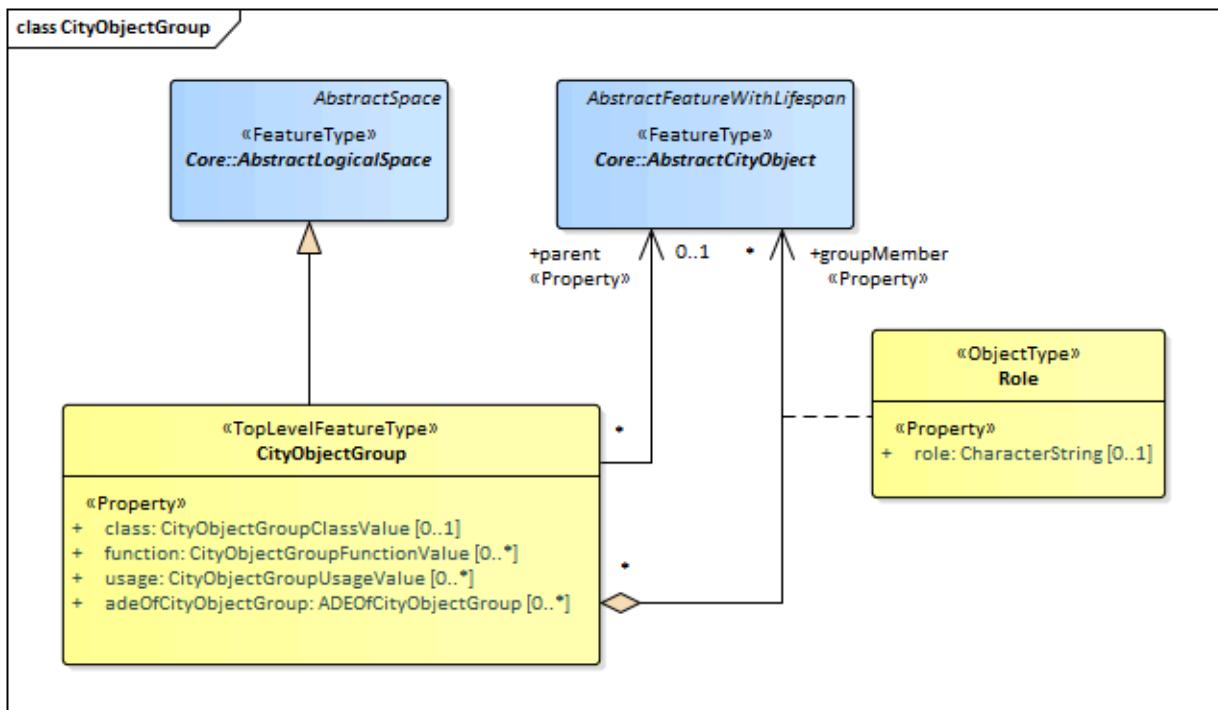


Figure 31 – UML diagram of the City Object Group Model.

The ADE data types provided for the CityObjectGroup module are illustrated in Figure 32.

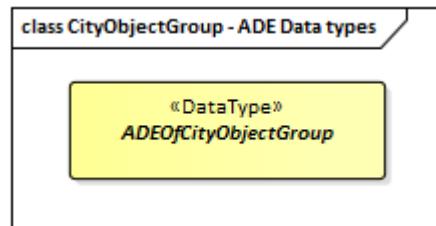


Figure 32 – ADE classes of the CityGML CityObjectGroup module.

The Code Lists provided for the CityObjectGroup module are illustrated in Figure 33.

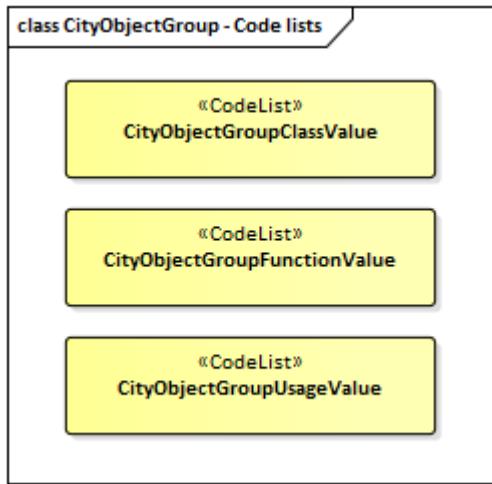


Figure 33 – Codelists from the CityGML CityObjectGroup module.

Table 21 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the CityObjectGroup module:

Table 21 – CityObjectGroup space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
CityObjectGroup	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface Possible classes from ADEs

7.5.1. Requirements

The following requirement defines the rules governing implementation of the CityGML City Object Group Module as an Implementation Specification.

REQUIREMENT 10

For each UML class defined or referenced in the CityObjectGroup Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.

REQUIREMENT 10

D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 11

Table 21 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the CityObjectGroup module. An Implementation Specification SHALL only support the boundaries described in Table 21

The use of extension capabilities by City Object Group elements is constrained by the following requirement:

REQUIREMENT 12

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.5.2. Class definitions

Table 22 – Classes defined in CityObjectGroup (ApplicationSchema)

NAME	DESCRIPTION
CityObjectGroup «TopLevelFeatureType»	A CityObjectGroup represents an application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group.
Role «ObjectType»	Role qualifies the function of a city object within the CityObjectGroup.

Table 23 – Data types defined in CityObjectGroup (ApplicationSchema)

NAME	DESCRIPTION
ADEOfCityObjectGroup «DataType»	ADEOfCityObjectGroup acts as a hook to define properties within an ADE that are to be added to a CityObjectGroup.

Table 24 – Code list classes defined in CityObjectGroup (ApplicationSchema)

NAME	DESCRIPTION
CityObjectGroupClassValue «CodeList»	CityObjectGroupClassValue is a code list used to further classify a CityObjectGroup.
CityObjectGroupFunction Value «CodeList»	CityObjectGroupFunctionValue is a code list that enumerates the different purposes of a CityObjectGroup.
CityObjectGroupUsage Value «CodeList»	CityObjectGroupUsageValue is a code list that enumerates the different uses of a CityObjectGroup.

7.5.3. Additional Information

Additional information about the CityObjectGroup Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.6. Dynamizer

REQUIREMENTS CLASS 5

Target type	Implementation Specification
Dependency	/req/req-class-core

The Dynamizer module provides the concepts that enable representation of time-varying data for city object properties as well as for integrating sensors with 3D city models. Dynamizers are objects that inject timeseries data for an individual attribute of the city object in which the Dynamizer is included. In order to represent dynamic (time-dependent) variations of its value, the timeseries data overrides the static value of the referenced city object attribute.

The dynamic values may be given by retrieving observation results directly from external sensor/IoT (Annex B.1.2.29) services using a sensor connection (e.g., OGC SensorThings API, Sensor Observation Service, or other sensor data platforms including MQTT [OASIS MQTT]). Alternatively, the dynamic values may be provided as atomic timeseries that represent time-varying data of a specific data type for a single contiguous time interval. The data can be provided in:

- external tabulated files, such as CSV or Excel sheets,
- external files that format timeseries data according to the OGC TimeseriesML Standard or the OGC Observations & Measurements standards, or
- inline as embedded time-value-pairs.

Furthermore, timeseries data can also be aggregated to form composite timeseries with non-overlapping time intervals.

By using the Dynamizer module, fast changes over a short or longer time period with respect to cities and city models can be represented. This includes:

- variations of spatial properties such as change of a feature's geometry, both in respect to shape and to location (e.g., moving objects),
- variations of thematic attributes such as changes of physical quantities like energy demands, temperatures, solar irradiation, traffic density, pollution concentration, or overpressure on building walls, and
- variations with respect to sensor or real-time data resulting from simulations or measurements.

The UML diagram of the Dynamizer module is depicted in Figure 34. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

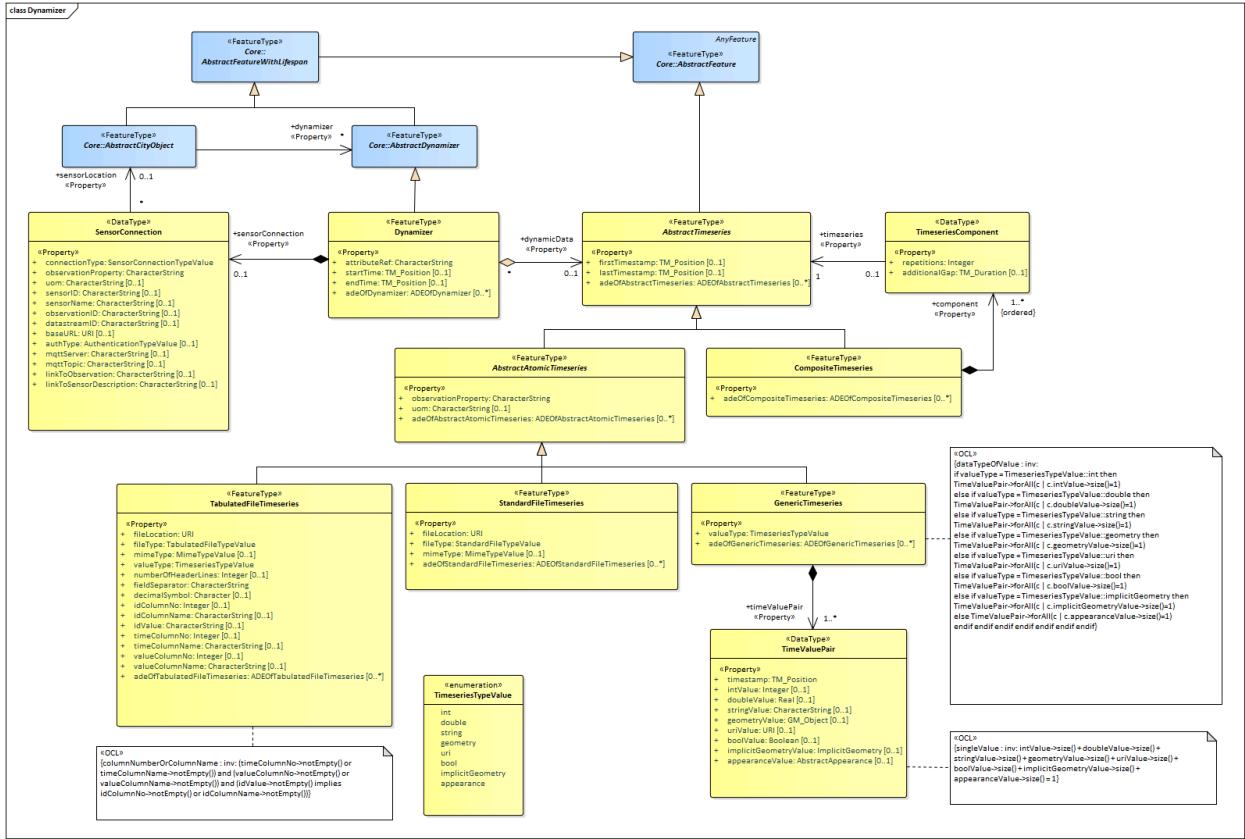


Figure 34 – UML diagram of the Dynamizer Model.

The ADE data types provided for the Dynamizer module are illustrated in Figure 35.

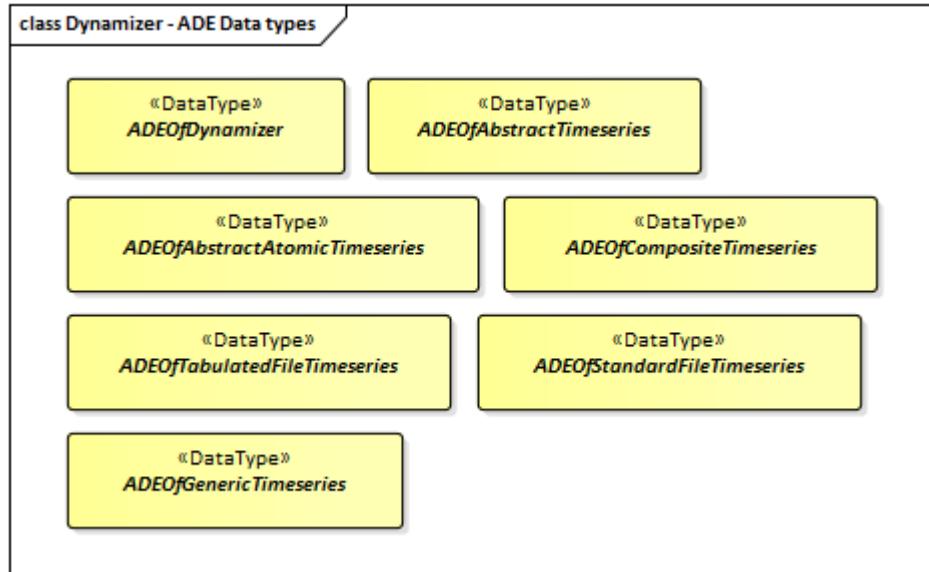


Figure 35 – ADE classes of the CityGML Dynamizer module.

The Code Lists provided for the Dynamizer module are illustrated in Figure 36.

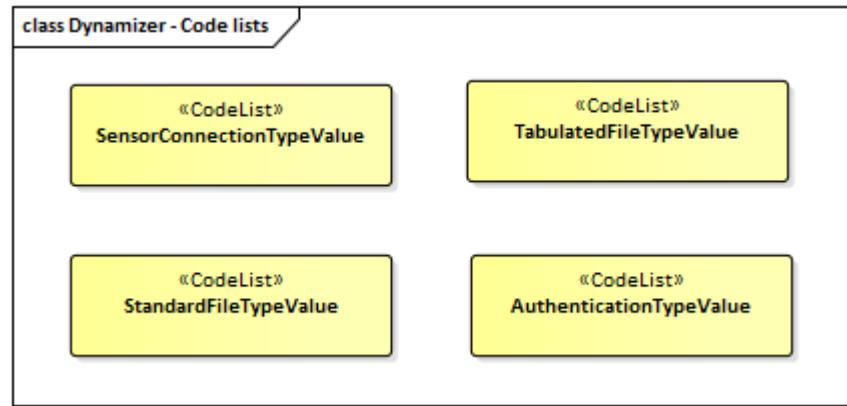


Figure 36 – Codelists from the CityGML Dynamizer module.

7.6.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Dynamizer Module as an Implementation Specification.

REQUIREMENT 13

For each UML class defined or referenced in the Dynamizer Package:

- A** The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
- B** The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
- C** The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
- D** The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
- E** The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
- F** The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Dynamizer elements is constrained by the following requirement:

REQUIREMENT 14

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.6.2. Class definitions

Table 25 – Classes defined in Dynamizer (ApplicationSchema)

NAME	DESCRIPTION
AbstractAtomicTimeseries «FeatureType»	AbstractAtomicTimeseries represents the attributes and relationships that are common to all kinds of atomic timeseries (GenericTimeseries, TabulatedFileTimeseries, StandardFileTimeseries). An atomic timeseries represents time-varying data of a specific data type for a single contiguous time interval.
AbstractTimeseries «FeatureType»	AbstractTimeseries is the abstract superclass representing any type of timeseries data.
CompositeTimeseries «FeatureType»	A CompositeTimeseries is a (possibly recursive) aggregation of atomic and composite timeseries. The components of a composite timeseries must have non-overlapping time intervals.
Dynamizer «FeatureType»	A Dynamizer is an object that injects timeseries data for an individual attribute of the city object in which it is included. The timeseries data overrides the static value of the referenced city object attribute in order to represent dynamic (time-dependent) variations of its value.
GenericTimeseries «Feature Type»	A GenericTimeseries represents time-varying data in the form of embedded time-value-pairs of a specific data type for a single contiguous time interval.
StandardFileTimeseries «FeatureType»	A StandardFileTimeseries represents time-varying data for a single contiguous time interval. The data is provided in an external file referenced in the StandardFileTimeseries. The data within the external file is encoded according to a dedicated format for the representation of timeseries data such as using the OGC TimeseriesML or OGC Observations & Measurements Standard. The data type of the data has to be specified within the external file.
TabulatedFileTimeseries «FeatureType»	A TabulatedFileTimeseries represents time-varying data of a specific data type for a single contiguous time interval. The data is provided in an external file referenced in the TabulatedFileTimeseries. The file contains table structured data using an appropriate file format such as comma-separated values (CSV), Microsoft Excel (XLSX) or Google Spreadsheet. The timestamps and the values are given in specific columns of the table. Each row represents a single time-value-pair. A subset of rows can be selected using the idColumn and idValue attributes.

Table 26 – Data types defined in Dynamizer (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractAtomicTimeseries «DataType»	ADEOfAbstractAtomicTimeseries acts as a hook to define properties within an ADE that are to be added to AbstractAtomicTimeseries.
ADEOfAbstractTimeseries «DataType»	ADEOfAbstractTimeseries acts as a hook to define properties within an ADE that are to be added to AbstractTimeseries.
ADEOfCompositeTimeseries «DataType»	ADEOfCompositeTimeseries acts as a hook to define properties within an ADE that are to be added to a CompositeTimeseries.
ADEOfDynamizer «DataType»	ADEOfDynamizer acts as a hook to define properties within an ADE that are to be added to a Dynamizer.
ADEOfGenericTimeseries «DataType»	ADEOfGenericTimeseries acts as a hook to define properties within an ADE that are to be added to a GenericTimeseries.
ADEOfStandardFileTimeseries «DataType»	ADEOfStandardFileTimeseries acts as a hook to define properties within an ADE that are to be added to a StandardFileTimeseries.
ADEOfTabulatedFileTimeseries «DataType»	ADEOfTabulatedFileTimeseries acts as a hook to define properties within an ADE that are to be added to a TabulatedFileTimeseries.
SensorConnection «DataType»	A SensorConnection provides all details that are required to retrieve a specific datastream from an external sensor web service. This data type comprises the service type (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms), the URL of the sensor service, the identifier for the sensor or thing, and its observed property as well as information about the required authentication method.
TimeseriesComponent «DataType»	TimeseriesComponent represents an element of a CompositeTimeseries.
TimeValuePair «DataType»	A TimeValuePair represents a value that is valid for a given timepoint. For each TimeValuePair, only one of the value properties can be used mutually exclusive. Which value property has to be provided depends on the selected value type in the GenericTimeSeries feature, in which the TimeValuePair is included.

Table 27 – Enumerated classes defined in Dynamizer (ApplicationSchema)

NAME	DESCRIPTION
TimeseriesTypeValue «Enumeration»	TimeseriesTypeValue enumerates the possible value types for GenericTimeseries and TimeValuePair.

Table 28 – Code list classes defined in Dynamizer (ApplicationSchema)

NAME	DESCRIPTION
AuthenticationTypeValue «CodeList»	AuthenticationTypeValue is a code list used to specify the authentication method to be used to access the referenced sensor service. Each value provides enough information such that a software application could determine the required access credentials.

NAME	DESCRIPTION
SensorConnectionTypeValue «CodeList»	SensorConnectionTypeValue is a code list used to specify the type of the referenced sensor service. Each value provides enough information such that a software application would be able to identify the API type and version.
StandardFileTypeValue «CodeList»	StandardFileTypeValue is a code list used to specify the type of the referenced external timeseries data file. Each value provides information about the standard and version.
TabulatedFileTypeValue «CodeList»	TabulatedFileTypeValue is a code list used to specify the data format of the referenced external tabulated data file.

7.6.3. Additional Information

Additional information about the Dynamizer Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.7. Generics

REQUIREMENTS CLASS 6

Target type	Implementation Specification
Dependency	/req/req-class-core

The Generics module provides the representation of generic city objects. These are city objects that are not covered by any explicitly modeled thematic class within the CityGML Conceptual Model. The Generics module also provides the representation of generic attributes which are attributes that are not explicitly represented in the CityGML Conceptual Model. In order to avoid problems concerning semantic interoperability, generic city objects and generic attributes shall only be used if appropriate thematic classes and attributes are not provided by any other CityGML module.

In accordance with the CityGML Space concept defined in the Core module (Clause 7.2) generic city objects can be represented as generic logical spaces, generic occupied spaces, generic unoccupied spaces, and generic thematic surfaces. In this way, spaces and surfaces can be defined that are not represented by any explicitly modeled class within CityGML that is a subclass of the classes *AbstractLogicalSpace*, *AbstractOccupiedSpace*, *AbstractUnoccupiedSpace* or *AbstractThematicSurface*, respectively. Generic city objects are represented in the UML model by the top-level feature types *GenericLogicalSpace*, *GenericOccupiedSpace*, *GenericUnoccupiedSpace* and *GenericThematicSurface*.

Generic attributes are defined as name-value pairs and are always associated with a city object. Generic attributes can be of type String, Integer, Double, Date, URI, Measure, and Code. In addition, generic attributes can be grouped under a common name as generic attribute sets.

The UML diagram of the Generics module is depicted in Figure 37. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

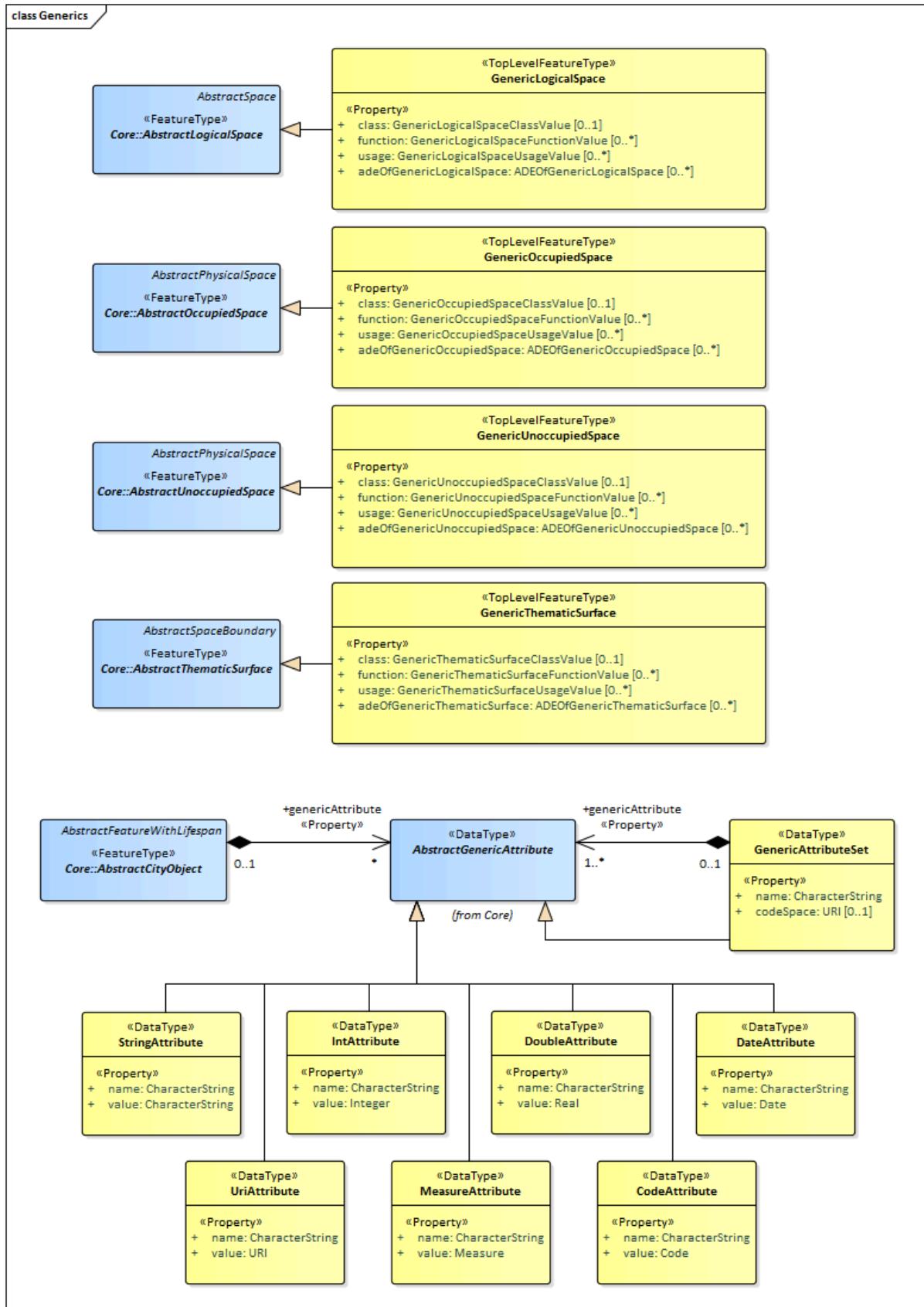


Figure 37 – UML diagram of the Generics Model.

The ADE data types provided for the Generics module are illustrated in Figure 38.

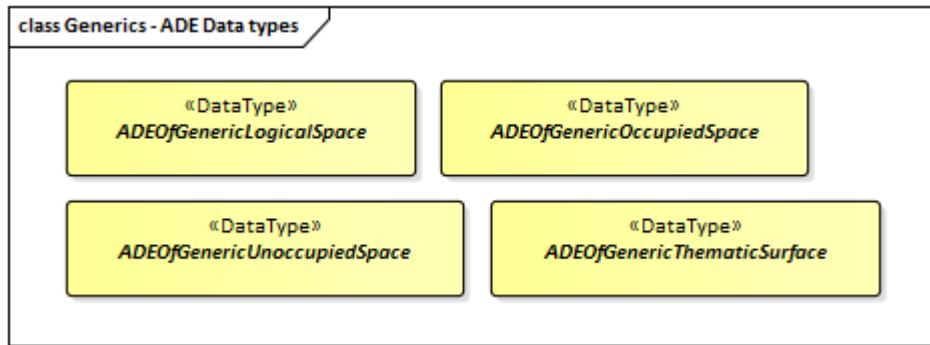


Figure 38 – ADE classes of the CityGML Generics module.

The Code Lists provided for the Generics module are illustrated in Figure 39.

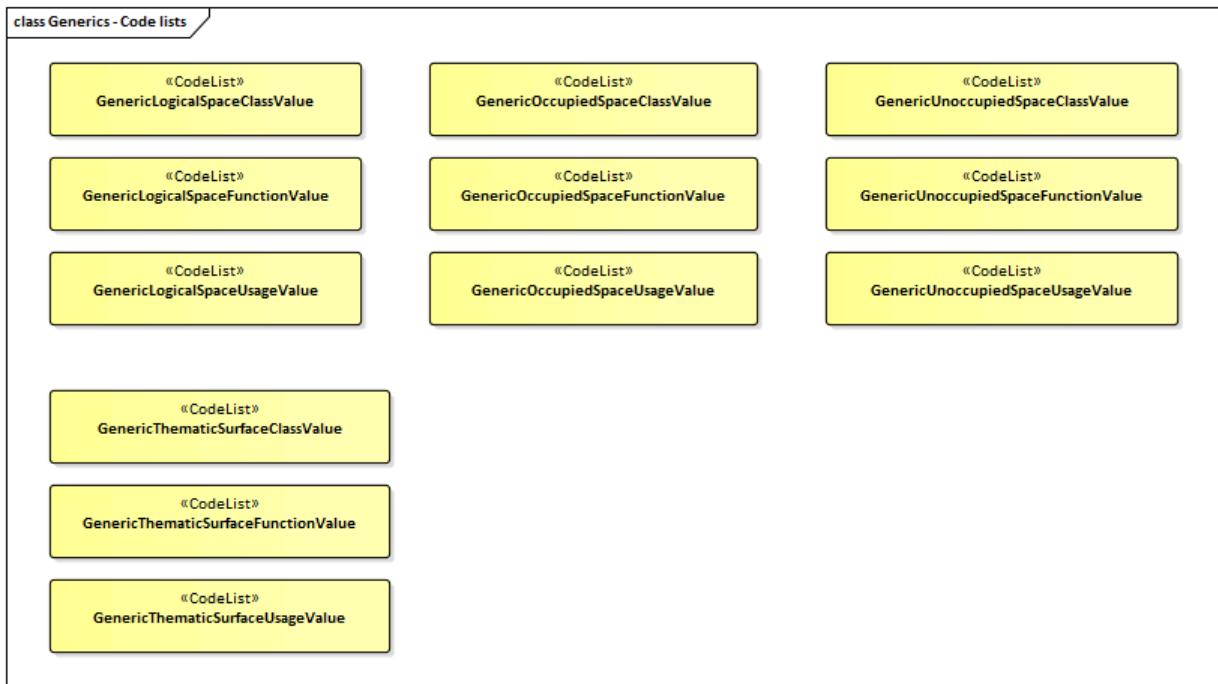


Figure 39 – Codelists from the CityGML Generics module.

Table 29 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Generics module:

Table 29 – Generics space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
GenericLogicalSpace	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
GenericOccupiedSpace	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
GenericUnoccupiedSpace	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

7.7.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Generics Module as an Implementation Specification.

REQUIREMENT 15

For each UML class defined or referenced in the Generics Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 16

Table 29 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Generics module. An Implementation Specification SHALL only support the boundaries described in Table 29

The decision of whether or not to use Generics is constrained by the following requirement:

REQUIREMENT 17

Generic objects and attributes SHALL only be used if a more specific feature class or attribute is not available from the CityGML conceptual model.

The use of extension capabilities by Generics elements is constrained by the following requirement:

REQUIREMENT 18

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.7.2. Class definitions

Table 30 – Classes defined in Generics (ApplicationSchema)

NAME	DESCRIPTION
GenericLogicalSpace «TopLevelFeatureType»	A GenericLogicalSpace is a space that is not represented by any explicitly modelled AbstractLogicalSpace subclass within CityGML.
GenericOccupiedSpace «TopLevelFeatureType»	A GenericOccupiedSpace is a space that is not represented by any explicitly modelled AbstractOccupiedSpace subclass within CityGML.
GenericThematicSurface «TopLevelFeatureType»	A GenericThematicSurface is a surface that is not represented by any explicitly modelled AbstractThematicSurface subclass within CityGML.
GenericUnoccupiedSpace «TopLevelFeatureType»	A GenericUnoccupiedSpace is a space that is not represented by any explicitly modelled AbstractUnoccupiedSpace subclass within CityGML.

Table 31 – Data types defined in Generics (ApplicationSchema)

NAME	DESCRIPTION
ADEOfGenericLogicalSpace «DataType»	ADEOfGenericLogicalSpace acts as a hook to define properties within an ADE that are to be added to a GenericLogicalSpace.
ADEOfGenericOccupiedSpace «DataType»	ADEOfGenericOccupiedSpace acts as a hook to define properties within an ADE that are to be added to a GenericOccupiedSpace.
ADEOfGenericThematicSurface «DataType»	ADEOfGenericThematicSurface acts as a hook to define properties within an ADE that are to be added to a GenericThematicSurface.
ADEOfGenericUnoccupiedSpace «DataType»	ADEOfGenericUnoccupiedSpace acts as a hook to define properties within an ADE that are to be added to a GenericUnoccupiedSpace.
CodeAttribute «DataType»	CodeAttribute is a data type used to define generic attributes of type “Code”.

NAME	DESCRIPTION
DateAttribute «DataType»	DateAttribute is a data type used to define generic attributes of type “Date”.
DoubleAttribute «DataType»	DoubleAttribute is a data type used to define generic attributes of type “Double”.
GenericAttributeSet «DataType»	A GenericAttributeSet is a named collection of generic attributes.
IntAttribute «DataType»	IntAttribute is a data type used to define generic attributes of type “Integer”.
MeasureAttribute «DataType»	MeasureAttribute is a data type used to define generic attributes of type “Measure”.
StringAttribute «DataType»	StringAttribute is a data type used to define generic attributes of type “String”.
UriAttribute «DataType»	UriAttribute is a data type used to define generic attributes of type “URI”.

Table 32 – Code list classes defined in Generics (ApplicationSchema)

NAME	DESCRIPTION
GenericLogicalSpaceClassValue «CodeList»	GenericLogicalSpaceClassValue is a code list used to further classify a GenericLogicalSpace.
GenericLogicalSpaceFunctionValue «CodeList»	GenericLogicalSpaceFunctionValue is a code list that enumerates the different purposes of a GenericLogicalSpace.
GenericLogicalSpaceUsageValue «CodeList»	GenericLogicalSpaceUsageValue is a code list that enumerates the different uses of a GenericLogicalSpace.
GenericOccupiedSpaceClassValue «CodeList»	GenericOccupiedSpaceClassValue is a code list used to further classify a GenericOccupiedSpace.
GenericOccupiedSpaceFunctionValue «CodeList»	GenericOccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericOccupiedSpace.
GenericOccupiedSpaceUsageValue «CodeList»	GenericOccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericOccupiedSpace.
GenericThematicSurfaceClassValue «CodeList»	GenericThematicSurfaceClassValue is a code list used to further classify a GenericThematicSurface.
GenericThematicSurfaceFunctionValue «CodeList»	GenericThematicSurfaceFunctionValue is a code list that enumerates the different purposes of a GenericThematicSurface.
GenericThematicSurfaceUsageValue «CodeList»	GenericThematicSurfaceUsageValue is a code list that enumerates the different uses of a GenericThematicSurface.
GenericUnoccupiedSpaceClassValue «CodeList»	GenericUnoccupiedSpaceClassValue is a code list used to further classify a GenericUnoccupiedSpace.
GenericUnoccupiedSpaceFunctionValue «CodeList»	GenericUnoccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericUnoccupiedSpace.

NAME	DESCRIPTION
GenericUnoccupiedSpace UsageValue «CodeList»	GenericUnoccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericUnoccupiedSpace.

7.7.3. Additional Information

Additional information about the Generics Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.8. LandUse

REQUIREMENTS CLASS 7

Target type	Implementation Specification
Dependency	/req/req-class-core

The LandUse module defines objects that can be used to describe areas of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, or wetlands (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 concepts are interlinked and often mixed in practice. Land use objects in CityGML support 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. Land use objects are represented in the UML model by the top-level feature type *LandUse*, which is also the only class of the LandUse module.

The UML diagram of the LandUse module is depicted in Figure 40. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

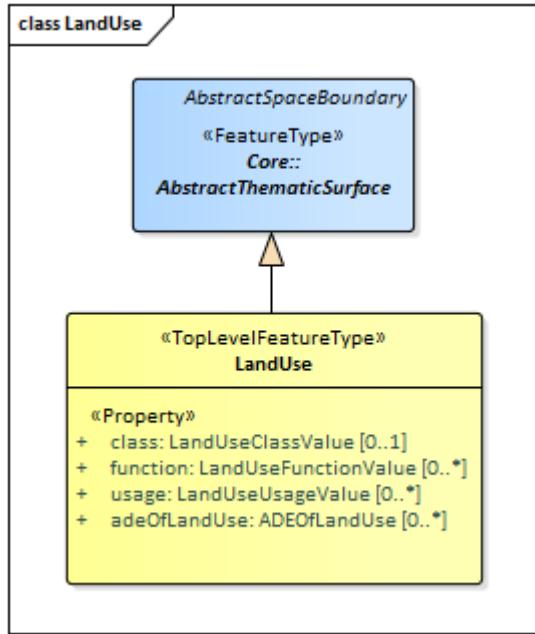


Figure 40 – UML diagram of the Land Use Model.

The ADE data types provided for the Land Use module are illustrated in Figure 41.

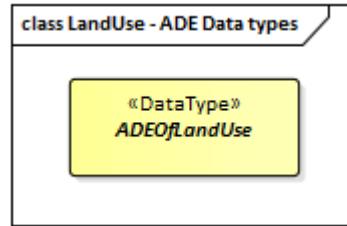


Figure 41 – ADE classes of the CityGML Land Use module.

The Code Lists provided for the Land Use module are illustrated in Figure 42.

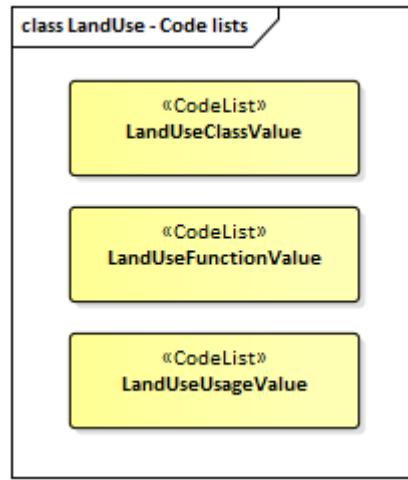


Figure 42 – Codelists from the CityGML Land Use module.

7.8.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Landuse Module as an Implementation Specification.

REQUIREMENT 19

For each UML class defined or referenced in the LandUse Package:

- A The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
- B The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
- C The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
- D The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
- E The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
- F The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Land Use elements is constrained by the following requirement:

REQUIREMENT 20

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.8.2. Class definitions

Table 33 – Classes defined in LandUse (ApplicationSchema)

NAME	DESCRIPTION
LandUse «TopLevelFeature Type»	A LandUse object is an area of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, or wetlands.

Table 34 – Data types defined in LandUse (ApplicationSchema)

NAME	DESCRIPTION
ADEOfLandUse «DataType»	ADEOfLandUse acts as a hook to define properties within an ADE that are to be added to a LandUse.

Table 35 – Code list classes defined in LandUse (ApplicationSchema)

NAME	DESCRIPTION
LandUseClassValue «Code List»	LandUseClassValue is a code list used to further classify a LandUse.
LandUseFunctionValue «CodeList»	LandUseFunctionValue is a code list that enumerates the different purposes of a Land Use.
LandUseUsageValue «Code List»	LandUseUsageValue is a code list that enumerates the different uses of a LandUse.

7.8.3. Additional Information

Additional information about the LandUse Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.9. PointCloud

REQUIREMENTS CLASS 8

Target type	Implementation Specification
Dependency	/req/req-class-core

The PointCloud module specifies how to encode the geometry of physical spaces and of thematic surfaces as 3D point clouds. In this way, the building hull, a room within a building or a single wall surface can be spatially represented by a point cloud only. The same applies to all other thematic feature types including transportation objects, vegetation, city furniture, etc. Point clouds can either be provided inline within a CityGML file or as reference to external point cloud files of common file types such as LAS or LAZ. Point clouds are represented in the UML model by the feature type *PointCloud*, which is also the only class of the PointCloud module.

The UML diagram of the PointCloud module is depicted in Figure 43. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

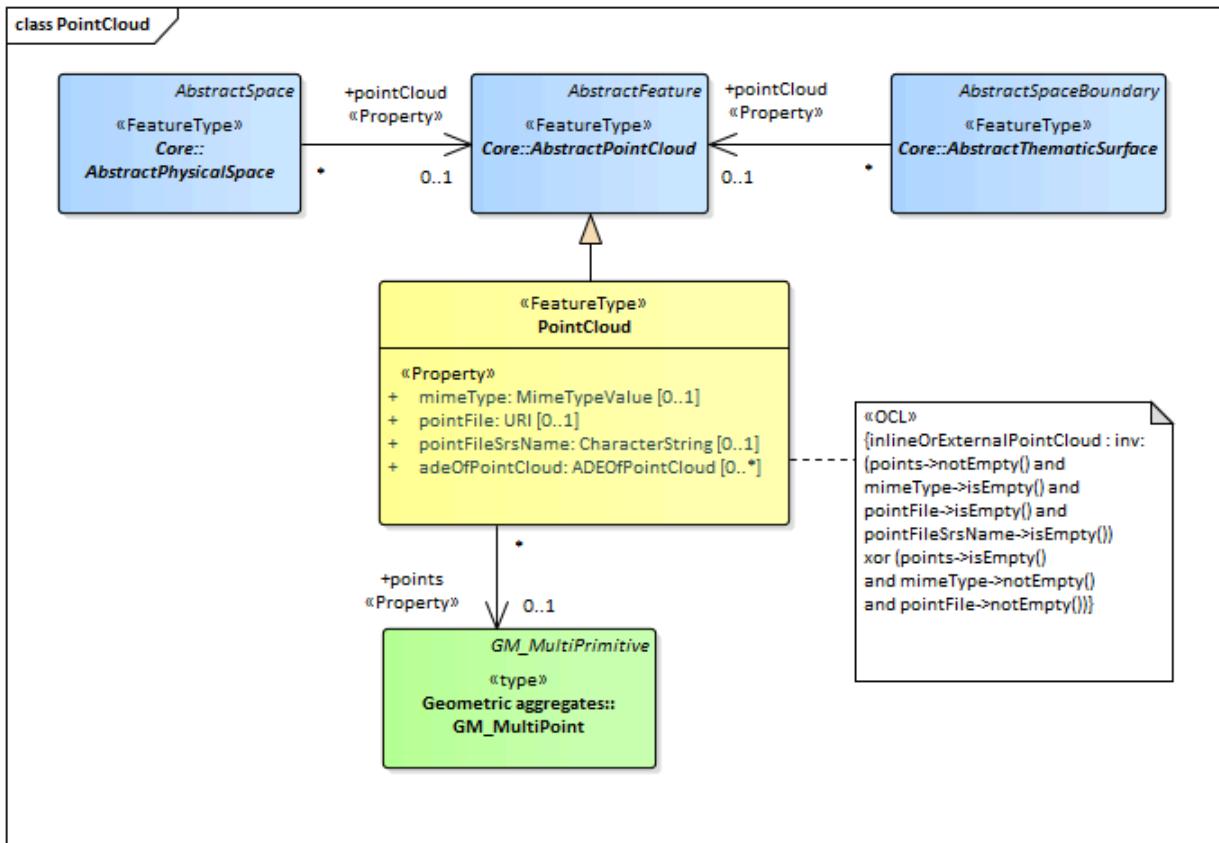


Figure 43 – UML diagram of the Point Cloud Model.

The ADE data types provided for the Point Cloud module are illustrated in Figure 44.

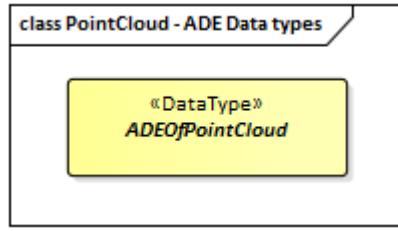


Figure 44 – ADE classes of the CityGML Point Cloud module.

7.9.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Pointcloud Module as an Implementation Specification.

REQUIREMENT 21

For each UML class defined or referenced in the PointCloud Package:

- A** The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
- B** The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
- C** The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
- D** The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
- E** The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
- F** The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Point Cloud elements is constrained by the following requirement:

REQUIREMENT 22

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.9.2. Class definitions

Table 36 – Classes defined in PointCloud (ApplicationSchema)

NAME	DESCRIPTION
PointCloud «FeatureType»	A PointCloud is an unordered collection of points that is a sampling of the geometry of a space or space boundary.

Table 37 – Data types defined in PointCloud (ApplicationSchema)

NAME	DESCRIPTION
ADEOfPointCloud «Data Type»	ADEOfPointCloud acts as a hook to define properties within an ADE that are to be added to a PointCloud.

7.9.3. Additional Information

Additional information about the PointCloud Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.10. Relief

REQUIREMENTS CLASS 9

Target type	Implementation Specification
Dependency	/req/req-class-core

The Relief module provides the representation of terrain which is an essential part of city models. In CityGML, the terrain is modeled by relief features. They are represented in the UML model by the top-level feature type *ReliefFeature*, which is the main class of the Relief module. The relief features, in turn, are collections of relief components that describe the Earth's surface, also known as the Digital Terrain Model. The relief components can have different terrain representations which can coexist. Each relief component may be specified as a regular raster or grid, as a TIN (Triangulated Irregular Network), by break lines, or by mass points. In addition, the validity of the relief components may be restricted to certain areas.

The UML diagram of the Relief module is depicted in Figure 45. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

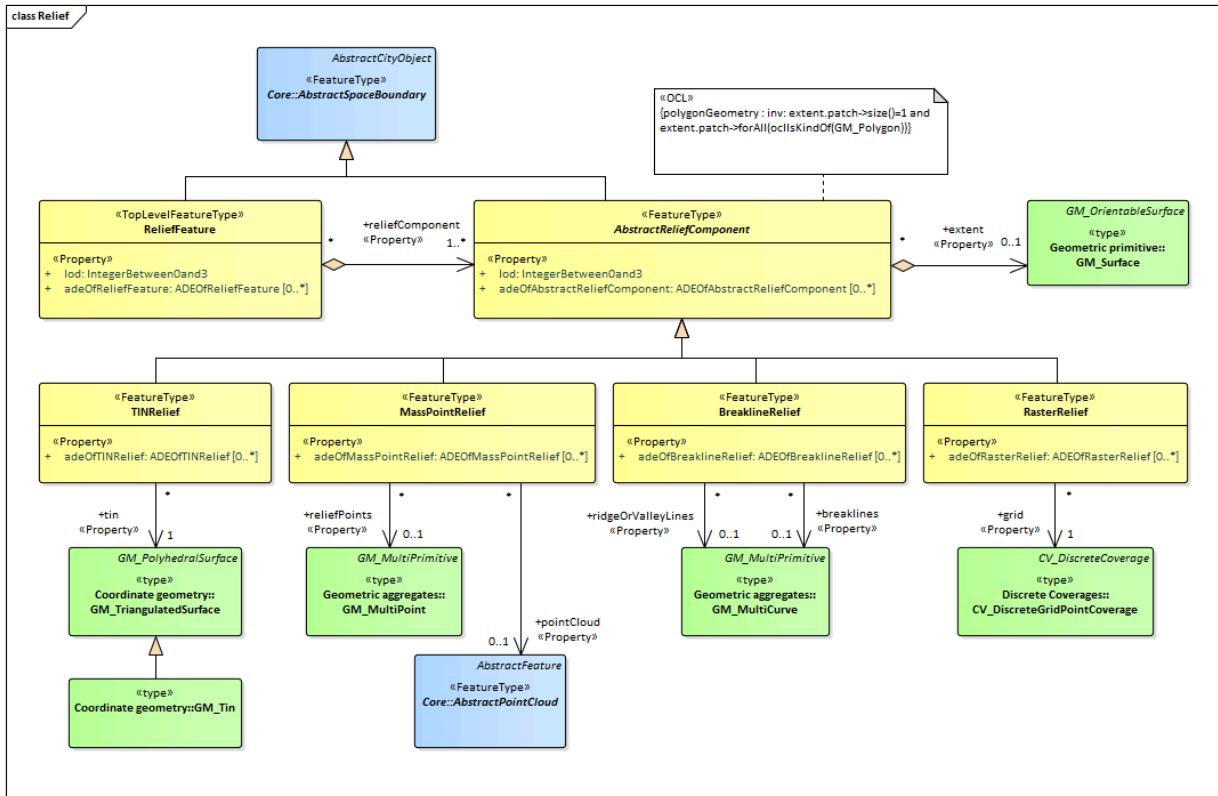


Figure 45 – UML diagram of Relief module.

The ADE data types provided for the Relief module are illustrated in Figure 46.

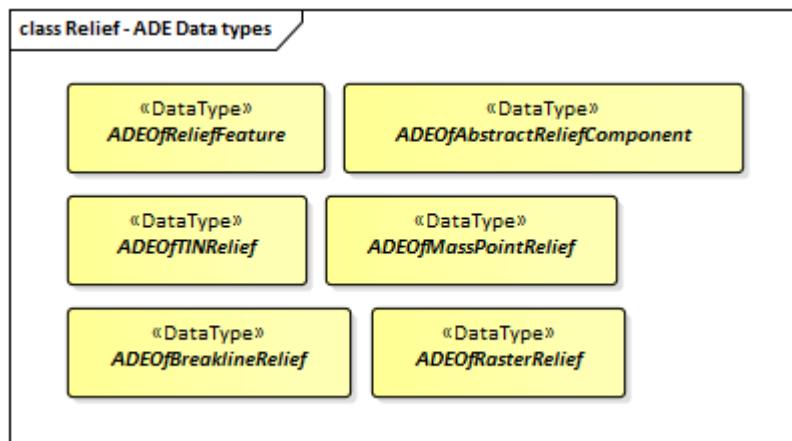


Figure 46 – ADE classes of the CityGML Relief module.

7.10.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Relief Module as an Implementation Specification.

REQUIREMENT 23

For each UML class defined or referenced in the Relief Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Relief elements is constrained by the following requirement:

REQUIREMENT 24

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.10.2. Class definitions

Table 38 – Classes defined in Relief (ApplicationSchema)

NAME	DESCRIPTION
AbstractReliefComponent «FeatureType»	An AbstractReliefComponent represents an element of the terrain surface – either a TIN, a raster or grid, mass points or break lines.
BreaklineRelief «Feature Type»	A BreaklineRelief represents a terrain component with 3D lines. These lines denote break lines or ridge/valley lines.
MassPointRelief «Feature Type»	A MassPointRelief represents a terrain component as a collection of 3D points.
RasterRelief «FeatureType»	A RasterRelief represents a terrain component as a regular raster or grid.

NAME	DESCRIPTION
ReliefFeature «TopLevel FeatureType»	A ReliefFeature is a collection of terrain components representing the Earth's surface, also known as the Digital Terrain Model.
TINRelief «FeatureType»	A TINRelief represents a terrain component as a triangulated irregular network.

Table 39 – Data types defined in Relief (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractRelief Component «DataType»	ADEOfAbstractReliefComponent acts as a hook to define properties within an ADE that are to be added to AbstractReliefComponent.
ADEOfBreaklineRelief «DataType»	ADEOfBreaklineRelief acts as a hook to define properties within an ADE that are to be added to a BreaklineRelief.
ADEOfMassPointRelief «DataType»	ADEOfMassPointRelief acts as a hook to define properties within an ADE that are to be added to a MassPointRelief.
ADEOfRasterRelief «DataType»	ADEOfRasterRelief acts as a hook to define properties within an ADE that are to be added to a RasterRelief.
ADEOfReliefFeature «DataType»	ADEOfReliefFeature acts as a hook to define properties within an ADE that are to be added to a ReliefFeature.
ADEOfTINRelief «DataType»	ADEOfTINRelief acts as a hook to define properties within an ADE that are to be added to a TINRelief.

7.10.3. Additional Information

Additional information about the Relief Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.11. Transportation

REQUIREMENTS CLASS 10

Target type	Implementation Specification
Dependency	/req/req-class-core

The Transportation module defines central elements of the traffic infrastructure. This includes the transportation objects road, track, and square for the movement of vehicles, bicycles, and pedestrians, the transportation object railway for the movement of wheeled vehicles on rails, as well as the transportation object waterway for the movement of vessels upon or within water bodies. The transportation objects are represented in the UML model by the top-level

feature types *Road*, *Track*, *Square*, *Railway*, and *Waterway*, which are the main classes of the Transportation module. Transportation objects can be subdivided into sections, which can be regular road, track or railway legs, into intersection areas, and into roundabouts.

For each transportation object, traffic spaces and auxiliary traffic spaces can be provided, which are bounded at the bottom by traffic areas and auxiliary traffic areas, respectively. Traffic areas are elements that are important in terms of traffic usage, such as driving lanes, sidewalks, and cycle lanes, whereas auxiliary traffic areas describe further elements, such as curbstones, middle lanes, and green areas. The corresponding spaces define the free space above the areas. In addition, each traffic space can have an optional clearance space. The transportation objects can be represented in different levels of granularity, either as a single area, split up into individual lanes or even decomposed into individual (carriage)ways. Furthermore, holes in the surfaces of roads, tracks or squares, such as road damages, manholes or drains, can be represented including their corresponding boundary surfaces. In addition, markings for the structuring or restriction of traffic can be added to the transportation areas. Examples are road markings and markings related to railway or waterway traffic.

The UML diagram of the Transportation module is depicted in Figure 47. A detailed discussion of this Requirements Class can be found in the [CityGML 3.0 Users Guide](#).

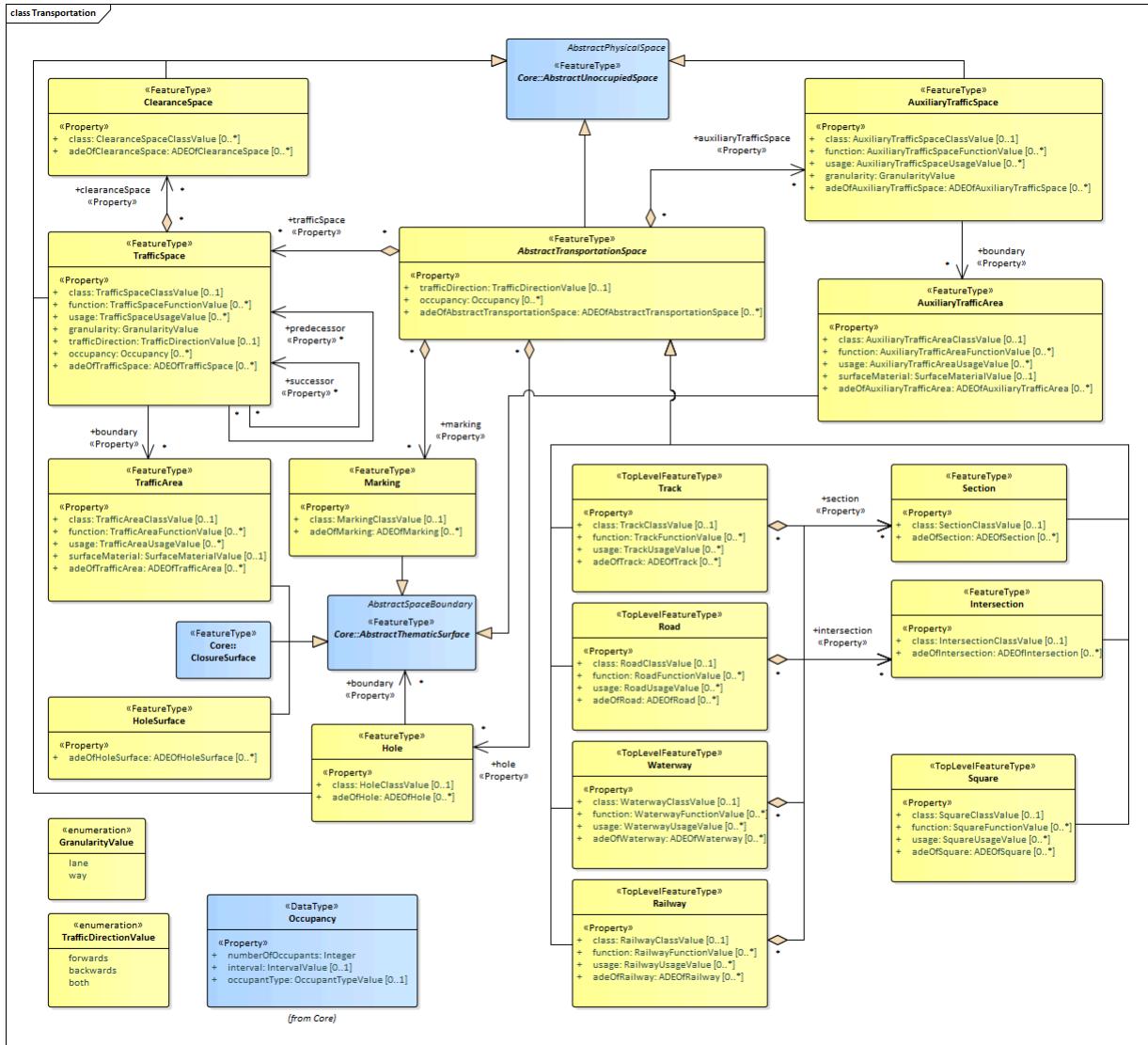


Figure 47 – UML diagram of the Transportation Model.

The ADE data types provided for the Transportation module are illustrated in Figure 48.

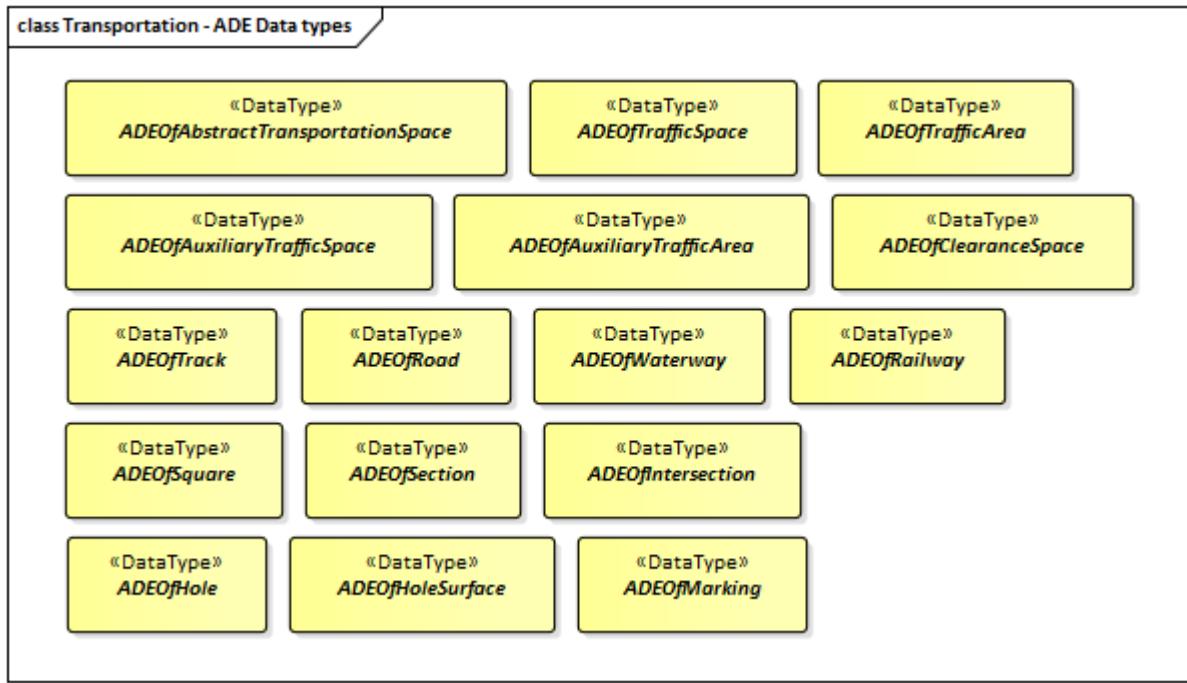


Figure 48 – ADE classes of the CityGML Transportation module.

The Code Lists provided for the Transportation module are illustrated in Figure 49.

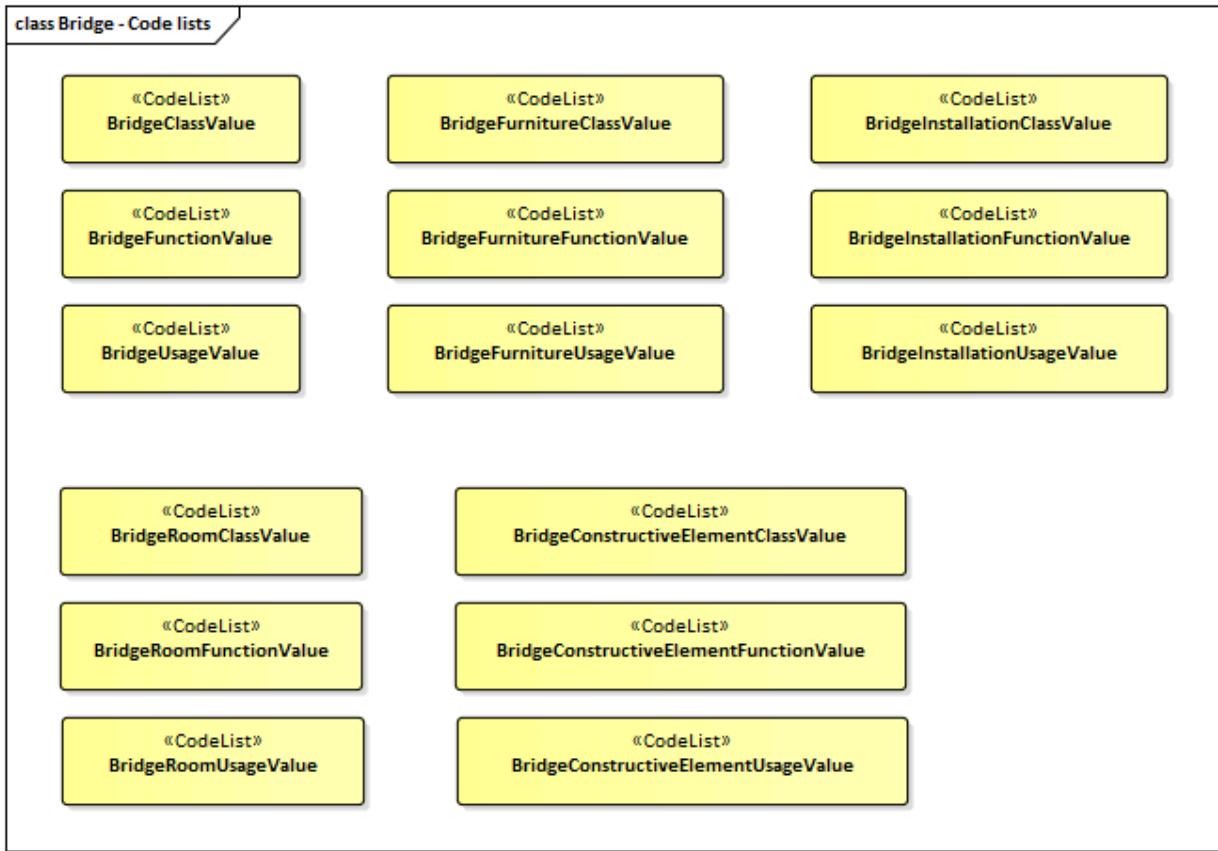


Figure 49 – Codelists from the CityGML Transportation module.

Table 40 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Transportation module:

Table 40 – Transportation space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractTransportationSpace	<ul style="list-style-type: none"> Transportation::Marking Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
AuxiliaryTrafficSpace	<ul style="list-style-type: none"> Transportation::AuxiliaryTrafficArea Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
ClearanceSpace	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs
Hole	<ul style="list-style-type: none"> Transportation::HoleSurface Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
Intersection	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Railway	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Road	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Section	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Square	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Track	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
TrafficSpace	<ul style="list-style-type: none"> • Transportation::TrafficArea • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Waterway	<ul style="list-style-type: none"> • Transportation::Marking • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

7.11.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Transportation Module as an Implementation Specification.

REQUIREMENT 25

For each UML class defined or referenced in the Transportation Package:

- A** The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.

REQUIREMENT 25

- B** The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
- C** The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
- D** The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
- E** The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
- F** The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 26

Table 40 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Transportation module. An Implementation Specification SHALL only support the boundaries described in Table 40

The use of extension capabilities by Transportation elements is constrained by the following requirement:

REQUIREMENT 27

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.11.2. Class definitions

Table 41 – Classes defined in Transportation (ApplicationSchema)

NAME	DESCRIPTION
AbstractTransportationSpace «FeatureType»	AbstractTransportationSpace is the abstract superclass of transportation objects such as Roads, Tracks, Railways, Waterways or Squares.
AuxiliaryTrafficArea «FeatureType»	An AuxiliaryTrafficArea is the ground surface of an AuxiliaryTrafficSpace.

NAME	DESCRIPTION
AuxiliaryTrafficSpace «FeatureType»	An AuxiliaryTrafficSpace is a space within the transportation space not intended for traffic purposes.
ClearanceSpace «FeatureType»	A ClearanceSpace represents the actual free space above a TrafficArea within which a mobile object can move without contacting an obstruction.
Hole «FeatureType»	A Hole is an opening in the surface of a Road, Track or Square such as road damages, manholes or drains. Holes can span multiple transportation objects.
HoleSurface «FeatureType»	A HoleSurface is a representation of the ground surface of a hole.
Intersection «FeatureType»	An Intersection is a transportation space that is a shared segment of multiple Road, Track, Railway, or Waterway objects (e.g. a crossing of two roads or a level crossing of a road and a railway).
Marking «FeatureType»	A Marking is a visible pattern on a transportation area relevant to the structuring or restriction of traffic. Examples are road markings and markings related to railway or waterway traffic.
Railway «TopLevelFeatureType»	A Railway is a transportation space used by wheeled vehicles on rails.
Road «TopLevelFeatureType»	A Road is a transportation space used by vehicles, bicycles and/or pedestrians.
Section «FeatureType»	A Section is a transportation space that is a segment of a Road, Railway, Track, or Waterway.
Square «TopLevelFeatureType»	A Square is a transportation space for unrestricted movement for vehicles, bicycles and/or pedestrians. This includes plazas as well as large sealed surfaces such as parking lots.
Track «TopLevelFeatureType»	A Track is a small path mainly used by pedestrians. Tracks can be segmented into Sections and Intersections.
TrafficArea «FeatureType»	A TrafficArea is the ground surface of a TrafficSpace. Traffic areas are the surfaces upon which traffic actually takes place.
TrafficSpace «FeatureType»	A TrafficSpace is a space in which traffic takes place. Traffic includes the movement of entities such as trains, vehicles, pedestrians, ships, or other transportation types.
Waterway «TopLevelFeatureType»	A Waterway is a transportation space used for the movement of vessels upon or within a water body.

Table 42 – Data types defined in Transportation (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractTransportationSpace «DataType»	ADEOfAbstractTransportationSpace acts as a hook to define properties within an ADE that are to be added to AbstractTransportationSpace.
ADEOfAuxiliaryTrafficArea «DataType»	ADEOfAuxiliaryTrafficArea acts as a hook to define properties within an ADE that are to be added to an AuxiliaryTrafficArea.

NAME	DESCRIPTION
ADEOfAuxiliaryTrafficSpace «DataType»	ADEOfAuxiliaryTrafficSpace acts as a hook to define properties within an ADE that are to be added to an AuxiliaryTrafficSpace.
ADEOfClearanceSpace «DataType»	ADEOfClearanceSpace acts as a hook to define properties within an ADE that are to be added to a ClearanceSpace.
ADEOfHole «DataType»	ADEOfHole acts as a hook to define properties within an ADE that are to be added to a Hole.
ADEOfHoleSurface «DataType»	ADEOfHoleSurface acts as a hook to define properties within an ADE that are to be added to a HoleSurface.
ADEOfIntersection «DataType»	ADEOfIntersection acts as a hook to define properties within an ADE that are to be added to an Intersection.
ADEOfMarking «DataType»	ADEOfMarking acts as a hook to define properties within an ADE that are to be added to a Marking.
ADEOfRailway «DataType»	ADEOfRailway acts as a hook to define properties within an ADE that are to be added to a Railway.
ADEOfRoad «DataType»	ADEOfRoad acts as a hook to define properties within an ADE that are to be added to a Road.
ADEOfSection «DataType»	ADEOfSection acts as a hook to define properties within an ADE that are to be added to a Section.
ADEOfSquare «DataType»	ADEOfSquare acts as a hook to define properties within an ADE that are to be added to a Square.
ADEOfTrack «DataType»	ADEOfTrack acts as a hook to define properties within an ADE that are to be added to a Track.
ADEOfTrafficArea «DataType»	ADEOfTrafficArea acts as a hook to define properties within an ADE that are to be added to a TrafficArea.
ADEOfTrafficSpace «DataType»	ADEOfTrafficSpace acts as a hook to define properties within an ADE that are to be added to a TrafficSpace.
ADEOfWaterway «DataType»	ADEOfWaterway acts as a hook to define properties within an ADE that are to be added to a Waterway.

Table 43 – Enumerated classes defined in Transportation (ApplicationSchema)

NAME	DESCRIPTION
GranularityValue «Enumeration»	GranularityValue enumerates the different levels of granularity in which transportation objects are represented.
TrafficDirectionValue «Enumeration»	TrafficDirectionValue enumerates the allowed directions of travel of a mobile object.

Table 44 – Code list classes defined in Transportation (ApplicationSchema)

NAME	DESCRIPTION
AuxiliaryTrafficAreaClassValue «CodeList»	AuxiliaryTrafficAreaClassValue is a code list used to further classify an AuxiliaryTrafficArea.
AuxiliaryTrafficAreaFunctionValue «CodeList»	AuxiliaryTrafficAreaFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficArea.
AuxiliaryTrafficAreaUsageValue «CodeList»	AuxiliaryTrafficAreaUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficArea.
AuxiliaryTrafficSpaceClassValue «CodeList»	AuxiliaryTrafficSpaceClassValue is a code list used to further classify an AuxiliaryTrafficSpace.
AuxiliaryTrafficSpaceFunctionValue «CodeList»	AuxiliaryTrafficSpaceFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficSpace.
AuxiliaryTrafficSpaceUsageValue «CodeList»	AuxiliaryTrafficSpaceUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficSpace.
ClearanceSpaceClassValue «CodeList»	ClearanceSpaceClassValue is a code list used to further classify a ClearanceSpace.
HoleClassValue «CodeList»	HoleClassValue is a code list used to further classify a Hole.
IntersectionClassValue «CodeList»	IntersectionClassValue is a code list used to further classify an Intersection.
MarkingClassValue «CodeList»	MarkingClassValue is a code list used to further classify a Marking.
RailwayClassValue «CodeList»	RailwayClassValue is a code list used to further classify a Railway.
RailwayFunctionValue «CodeList»	RailwayFunctionValue is a code list that enumerates the different purposes of a Railway.
RailwayUsageValue «CodeList»	RailwayUsageValue is a code list that enumerates the different uses of a Railway.
RoadClassValue «CodeList»	RoadClassValue is a code list used to further classify a Road.
RoadFunctionValue «CodeList»	RoadFunctionValue is a code list that enumerates the different purposes of a Road.
RoadUsageValue «CodeList»	RoadUsageValue is a code list that enumerates the different uses of a Road.
SectionClassValue «CodeList»	SectionClassValue is a code list used to further classify a Section.
SquareClassValue «CodeList»	SquareClassValue is a code list used to further classify a Square.
SquareFunctionValue «CodeList»	SquareFunctionValue is a code list that enumerates the different purposes of a Square.

NAME	DESCRIPTION
SquareUsageValue «CodeList»	SquareUsageValue is a code list that enumerates the different uses of a Square.
SurfaceMaterialValue «CodeList»	SurfaceMaterialValue is a code list that enumerates the different surface materials.
TrackClassValue «CodeList»	TrackClassValue is a code list used to further classify a Track.
TrackFunctionValue «CodeList»	TrackFunctionValue is a code list that enumerates the different purposes of a Track.
TrackUsageValue «CodeList»	TrackUsageValue is a code list that enumerates the different uses of a Track.
TrafficAreaClassValue «CodeList»	TrafficAreaClassValue is a code list used to further classify a TrafficArea.
TrafficAreaFunctionValue «CodeList»	TrafficAreaFunctionValue is a code list that enumerates the different purposes of a TrafficArea.
TrafficAreaUsageValue «CodeList»	TrafficAreaUsageValue is a code list that enumerates the different uses of a TrafficArea.
TrafficSpaceClassValue «CodeList»	TrafficSpaceClassValue is a code list used to further classify a TrafficSpace.
TrafficSpaceFunctionValue «CodeList»	TrafficSpaceFunctionValue is a code list that enumerates the different purposes of a TrafficSpace.
TrafficSpaceUsageValue «CodeList»	TrafficSpaceUsageValue is a code list that enumerates the different uses of a TrafficSpace.
WaterwayClassValue «CodeList»	WaterwayClassValue is a code list used to further classify a Waterway.
WaterwayFunctionValue «CodeList»	WaterwayFunctionValue is a code list that enumerates the different purposes of a Waterway.
WaterwayUsageValue «CodeList»	WaterwayUsageValue is a code list that enumerates the different uses of a Waterway.

7.11.3. Additional Information

Additional information about the Transportation Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.12. Vegetation

REQUIREMENTS CLASS 11

Target type	Implementation Specification
Dependency	/req/req-class-core

The Vegetation module defines the concepts to represent vegetation within city models. Vegetation can be represented either as solitary vegetation objects, such as trees, bushes and ferns, or as vegetation areas that are covered by plants of a given species or a typical mixture of plant species, such as forests, steppes and wet meadows. Vegetation is represented in the UML model by the top-level feature types *SolitaryVegetationObject* and *PlantCover*, which are also the only classes of the Vegetation module.

The UML diagram of the Vegetation module is depicted in Figure 50. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

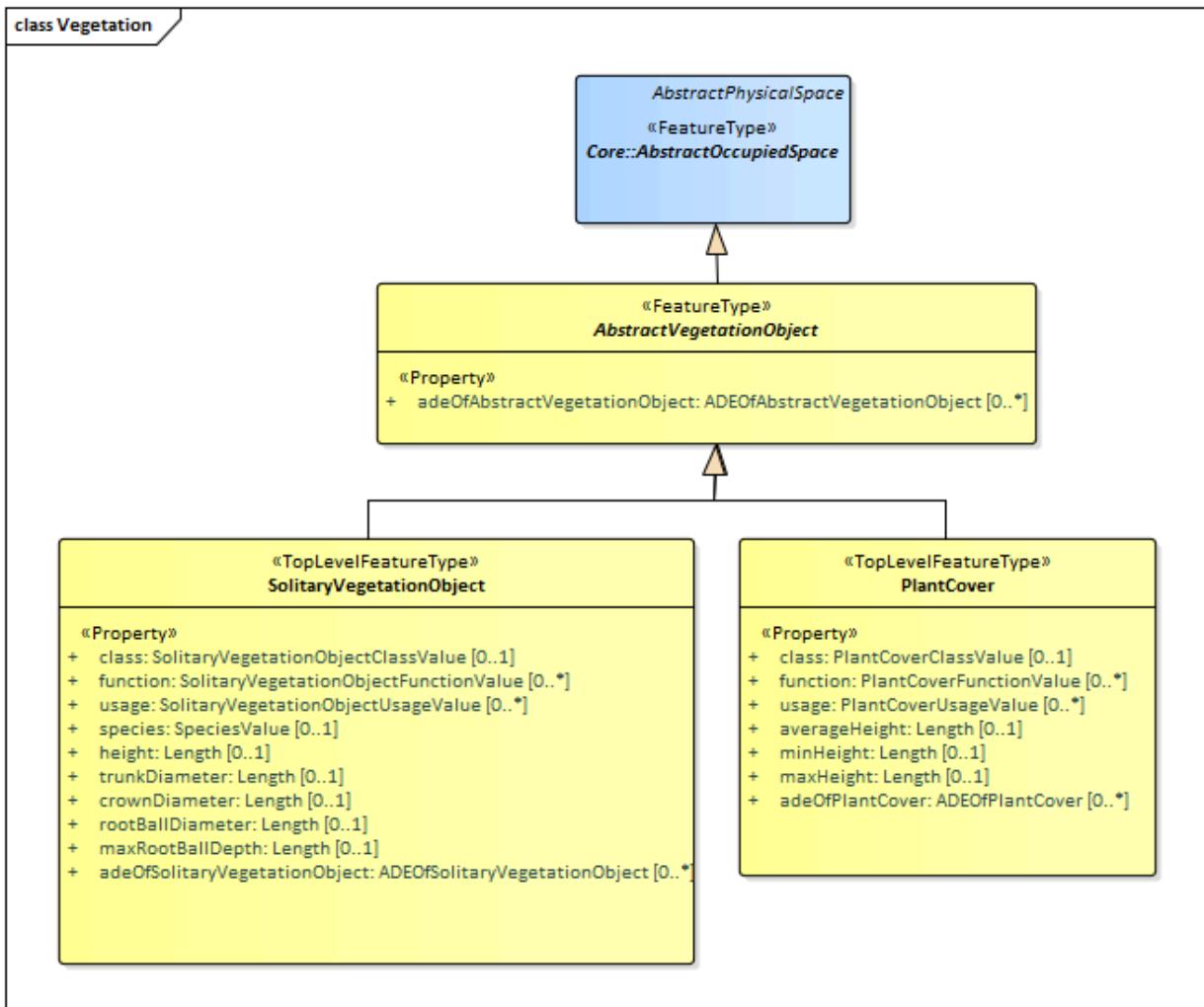


Figure 50 – UML diagram of the Vegetation Model.

The ADE data types provided for the Vegetation module are illustrated in Figure 51.

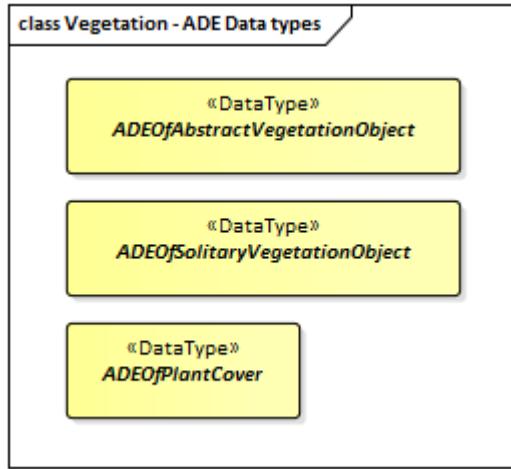


Figure 51 – ADE classes of the CityGML Vegetation module.

The Code Lists provided for the Vegetation module are illustrated in Figure 52.

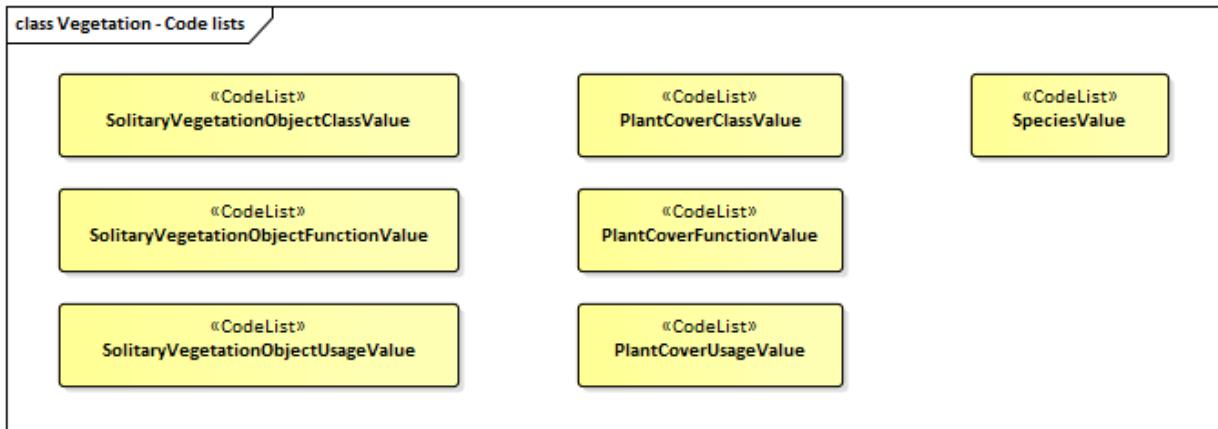


Figure 52 – Codelists from the CityGML Vegetation module.

Table 45 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Vegetation module.

Table 45 – Vegetation space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractVegetationObject	No boundaries allowed
PlantCover	No boundaries allowed
SolitaryVegetationObject	No boundaries allowed

7.12.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Vegetation Module as an Implementation Specification.

REQUIREMENT 28

For each UML class defined or referenced in the Vegetation Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 29

Table 45 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Vegetation module. An Implementation Specification SHALL only support the boundaries described in Table 45

The use of extension capabilities by Vegetation elements is constrained by the following requirement:

REQUIREMENT 30

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.12.2. Class definitions

Table 46 – Classes defined in Vegetation (ApplicationSchema)

NAME	DESCRIPTION
AbstractVegetationObject «FeatureType»	AbstractVegetationObject is the abstract superclass for all kinds of vegetation objects.
PlantCover «TopLevel FeatureType»	A PlantCover represents a space covered by vegetation.
SolitaryVegetationObject «TopLevelFeatureType»	A SolitaryVegetationObject represents individual vegetation objects, e.g. trees or bushes.

Table 47 – Data types defined in Vegetation (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractVegetation Object «DataType»	ADEOfAbstractVegetationObject acts as a hook to define properties within an ADE that are to be added to AbstractVegetationObject.
ADEOfPlantCover «Data Type»	ADEOfPlantCover acts as a hook to define properties within an ADE that are to be added to a PlantCover.
ADEOfSolitaryVegetation Object «DataType»	ADEOfSolitaryVegetationObject acts as a hook to define properties within an ADE that are to be added to a SolitaryVegetationObject.

Table 48 – Code list classes defined in Vegetation (ApplicationSchema)

NAME	DESCRIPTION
PlantCoverClassValue «CodeList»	PlantCoverClassValue is a code list used to further classify a PlantCover.
PlantCoverFunctionValue «CodeList»	PlantCoverFunctionValue is a code list that enumerates the different purposes of a PlantCover.
PlantCoverUsageValue «CodeList»	PlantCoverUsageValue is a code list that enumerates the different uses of a PlantCover.
SolitaryVegetationObject ClassValue «CodeList»	SolitaryVegetationObjectClassValue is a code list used to further classify a SolitaryVegetationObject.
SolitaryVegetationObject FunctionValue «CodeList»	SolitaryVegetationObjectFunctionValue is a code list that enumerates the different purposes of a SolitaryVegetationObject.
SolitaryVegetationObject UsageValue «CodeList»	SolitaryVegetationObjectUsageValue is a code list that enumerates the different uses of a SolitaryVegetationObject.

NAME	DESCRIPTION
SpeciesValue «CodeList»	A SpeciesValue is a code list that enumerates the species of a SolitaryVegetation Object.

7.12.3. Additional Information

Additional information about the Vegetation Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.13. Versioning

REQUIREMENTS CLASS 12

Target type	Implementation Specification
Dependency	/req/req-class-core

The Versioning module defines the concepts that enable representing multiple versions of a city model. A specific version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Each version can be complemented by version transitions that describe the change of the state of a city model from one version to another and that give the reason for the change and the modifications applied.

By using the Versioning module, slow changes over a long time period with respect to cities and city models can be represented. This includes the creation and termination of objects (e.g., construction or demolition of sites, planting of trees, construction of new roads), structural changes of objects (e.g., extension of buildings), and changes in the status of an object (e.g., change of building owner, change of the traffic direction of a road to a one-way street). In this way, the history or evolution of cities and city models can be modeled, parallel or alternative versions of cities and city models can be managed, and changes of geometries and thematic properties of individual city objects over time can be tracked.

The UML diagram of the Versioning module is depicted in Figure 53. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

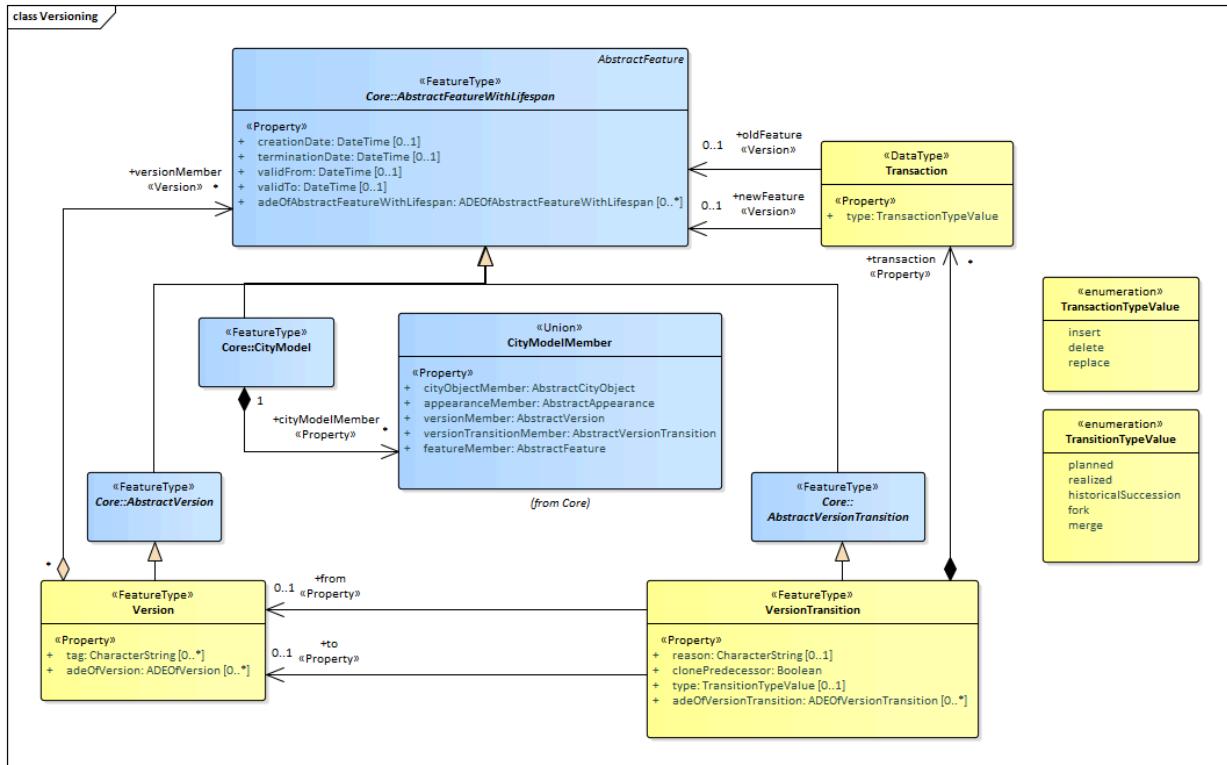


Figure 53 – UML diagram of the Versioning Model.

The ADE data types provided for the Versioning module are illustrated in Figure 54.

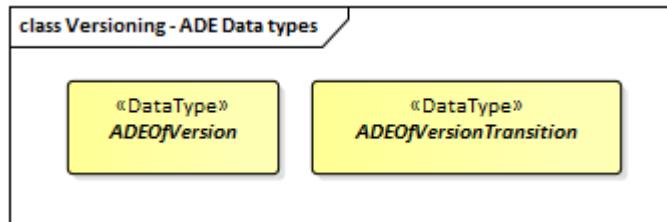


Figure 54 – ADE classes of the CityGML Versioning module.

7.13.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Versioning Module as an Implementation Specification.

REQUIREMENT 31

For each UML class defined or referenced in the Versioning Package:

A

The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.

REQUIREMENT 31

B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The use of extension capabilities by Versioning elements is constrained by the following requirement:

REQUIREMENT 32

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.13.2. Class definitions

Table 49 – Classes defined in Versioning (ApplicationSchema)

NAME	DESCRIPTION
Version «FeatureType»	Version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Versions can have names, a description and can be labeled with an arbitrary number of user defined tags.
VersionTransition «Feature Type»	VersionTransition describes the change of the state of a city model from one version to another. Version transitions can have names, a description and can be further qualified by a type and a reason.

Table 50 – Data types defined in Versioning (ApplicationSchema)

NAME	DESCRIPTION
ADEOfVersion «DataType»	ADEOfVersion acts as a hook to define properties within an ADE that are to be added to a Version.
ADEOfVersionTransition «DataType»	ADEOfVersionTransition acts as a hook to define properties within an ADE that are to be added to a VersionTransition.
Transaction «DataType»	Transaction represents a modification of the city model by the creation, termination, or replacement of a specific city object. While the creation of a city object also marks its first object version, the termination marks the end of existence of a real world object and, hence, also terminates the final version of a city object. The replacement of a city object means that a specific version of it is replaced by a new version.

Table 51 – Enumerated classes defined in Versioning (ApplicationSchema)

NAME	DESCRIPTION
TransactionTypeValue «Enumeration»	TransactionTypeValue enumerates the three possible types of transactions: insert, delete, or replace.
TransitionTypeValue «Enumeration»	TransitionTypeValue enumerates the different kinds of version transitions. “planned” and “fork” should be used in cases when from one city model version multiple successor versions are being created. “realized” and “merge” should be used when different city model versions are converging into a common successor version.

7.13.3. Additional Information

Additional information about the Versioning Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.14. WaterBody

REQUIREMENTS CLASS 13

Target type	Implementation Specification
Dependency	/req/req-class-core

The WaterBody module provides the representation of significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth. Examples of such water bodies that can be modeled with CityGML are rivers, canals, lakes, and basins. Water

bodies are represented in the UML model by the top-level feature type *WaterBody*, which is the main class of the *WaterBody* module.

Water bodies can be bounded by water surfaces, which represent the upper exterior interface between the water body and the atmosphere, and by water ground surfaces, which represent the exterior boundary surfaces of the submerged bottom of a water body (e.g., DTM or floor of a 3D basin object). Water surfaces are dynamic surfaces, thus, the visible water surface can regularly as well as irregularly change in height and covered area due to natural forces such as tides and floods.

The UML diagram of the WaterBody module is depicted in Figure 55. A detailed discussion of this Requirements Class can be found in the [OGC CityGML 3.0 Users Guide](#).

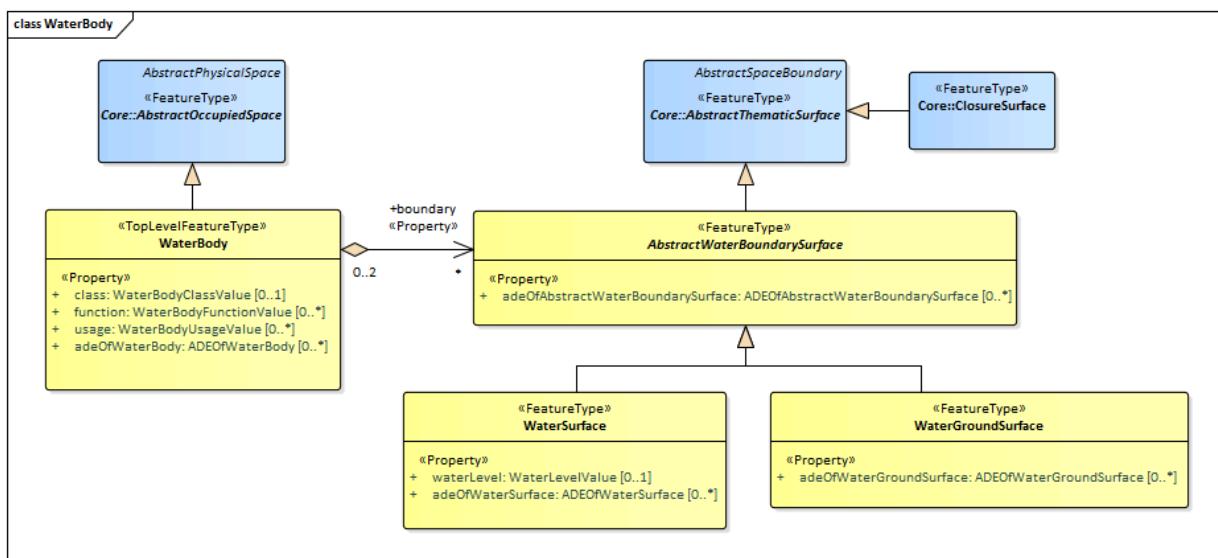


Figure 55 – UML diagram of the Water Body Model.

The ADE data types provided for the Water Body module are illustrated in Figure 56.

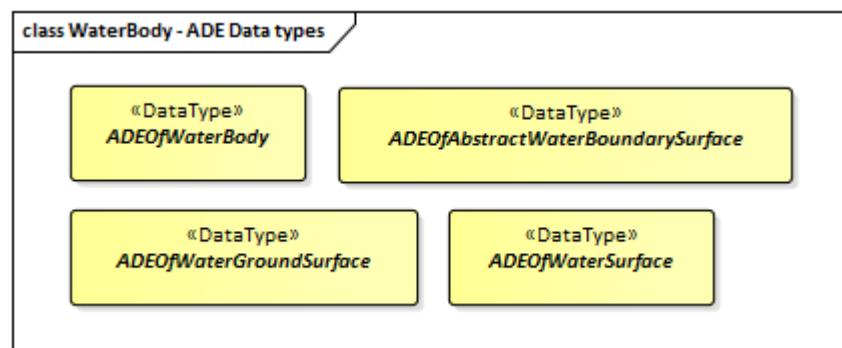


Figure 56 – ADE classes of the CityGML Water Body module.

The Code Lists provided for the Water Body module are illustrated in Figure 57.

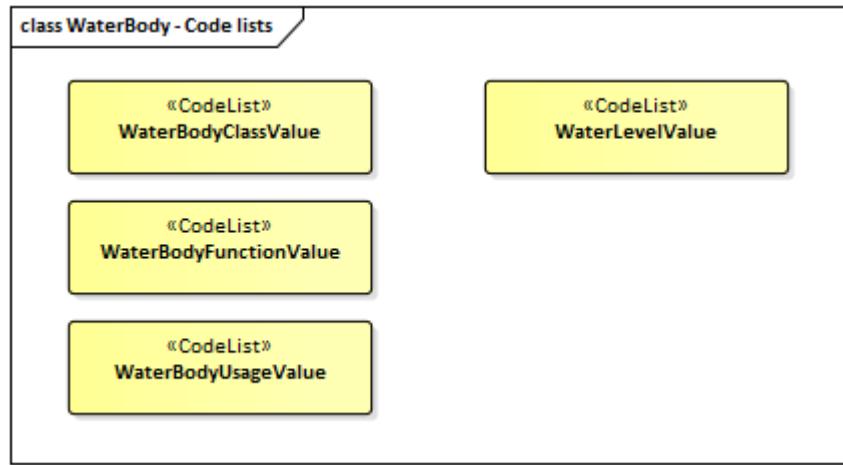


Figure 57 – Codelists from the CityGML Water Body module.

Table 52 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the WaterBody module:

Table 52 – WaterBody space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
WaterBody	<ul style="list-style-type: none"> • WaterBody::AbstractWaterBoundarySurface and all subclasses, i.e. <ul style="list-style-type: none"> • WaterBody::WaterGroundSurface, • WaterBody::WaterSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

7.14.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Waterbody Module as an Implementation Specification.

REQUIREMENT 33

For each UML class defined or referenced in the Waterbody Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.

REQUIREMENT 33

C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 34

Table 52 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Waterbody module. An Implementation Specification SHALL only support the boundaries described in Table 52

The use of extension capabilities by Waterbody elements is constrained by the following requirement:

REQUIREMENT 35

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.14.2. Class definitions

Table 53 – Classes defined in WaterBody (ApplicationSchema)

NAME	DESCRIPTION
AbstractWaterBoundarySurface «FeatureType»	AbstractWaterBoundarySurface is the abstract superclass for all kinds of thematic surfaces bounding a water body.
WaterBody «TopLevel FeatureType»	A WaterBody represents significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth.
WaterGroundSurface «FeatureType»	A WaterGroundSurface represents the exterior boundary surface of the submerged bottom of a water body.

NAME	DESCRIPTION
WaterSurface «Feature Type»	A WaterSurface represents the upper exterior interface between a water body and the atmosphere.

Table 54 – Data types defined in WaterBody (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractWaterBoundarySurface «Data Type»	ADEOfAbstractWaterBoundarySurface acts as a hook to define properties within an ADE that are to be added to AbstractWaterBoundarySurface.
ADEOfWaterBody «Data Type»	ADEOfWaterBody acts as a hook to define properties within an ADE that are to be added to a WaterBody.
ADEOfWaterGroundSurface «Data Type»	ADEOfWaterGroundSurface acts as a hook to define properties within an ADE that are to be added to a WaterGroundSurface.
ADEOfWaterSurface «Data Type»	ADEOfWaterSurface acts as a hook to define properties within an ADE that are to be added to a WaterSurface.

Table 55 – Code list classes defined in WaterBody (ApplicationSchema)

NAME	DESCRIPTION
WaterBodyClassValue «CodeList»	WaterBodyClassValue is a code list used to further classify a WaterBody.
WaterBodyFunctionValue «CodeList»	WaterBodyFunctionValue is a code list that enumerates the different purposes of a WaterBody.
WaterBodyUsageValue «CodeList»	WaterBodyUsageValue is a code list that enumerates the different uses of a WaterBody.
WaterLevelValue «CodeList»	WaterLevelValue is a code list that enumerates the different levels of a water surface.

7.14.3. Additional Information

Additional information about the WaterBody Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.15. Construction

REQUIREMENTS CLASS 14

Target type	Implementation Specification
Dependency	/req/req-class-core
Dependency	/req/req-class-generics

The Construction module defines concepts that are common to all forms of constructions. Constructions are objects that are manufactured by humans from construction materials, are connected to earth, and are intended to be permanent. The Construction module focuses on as-built representations of constructions and integrates all concepts that are similar over different types of constructions, in particular buildings, bridges, and tunnels. In addition, for representing man-made structures that are neither buildings, nor bridges, nor tunnels so-called other constructions (e.g., large chimneys or city walls) can be defined.

Furniture, installations, and constructive elements are further concepts that are defined in the Construction module. Installations are permanent parts of a construction that strongly affect the outer or inner appearance of the construction and that cannot be moved (e.g., balconies, chimneys, or stairs), whereas furniture represent moveable objects of a construction (e.g., tables and chairs). Constructive elements allow for decomposing a construction into volumetric components, such as walls, beams, and slabs. Constructions and constructive elements can be bounded by different types of surfaces. In this way, the outer structure of constructions and constructive elements can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of interior spaces can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of constructions, i.e., windows and doors, can be represented as so-called filling elements including their corresponding filling surfaces.

The UML diagram of the Construction module is depicted in Figure 58. The Construction module defines concepts that are inherited and, where necessary, are specialized by the modules Building, Bridge, and Tunnel (cf. Clause 7.17, Clause 7.16, and Clause 7.18). A detailed discussion of the Requirements Class Construction can be found in the [CityGML 3.0 Users Guide](#).

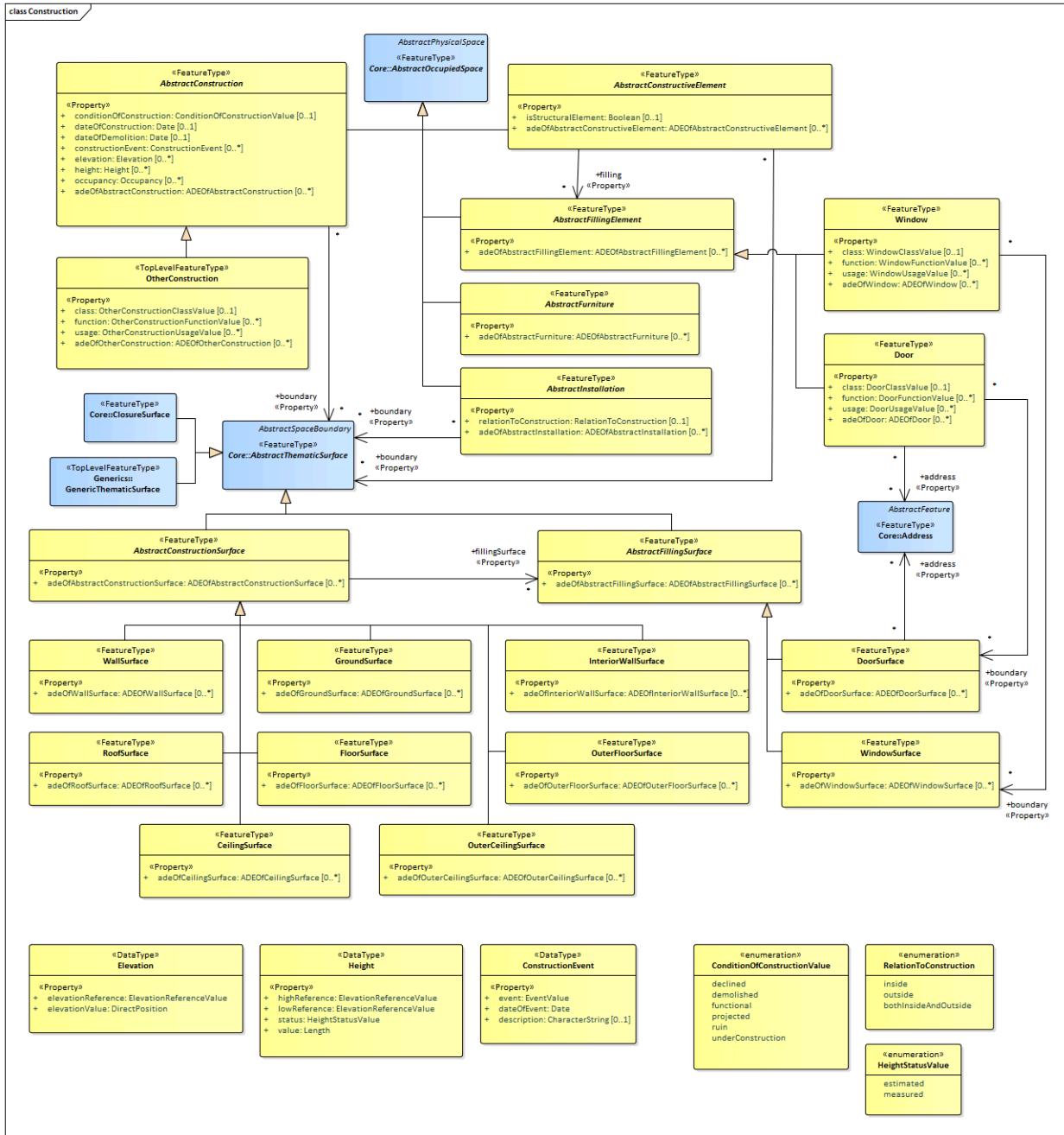


Figure 58 – UML diagram of the Construction Model.

The ADE data types provided for the Construction module are illustrated in Figure 59.

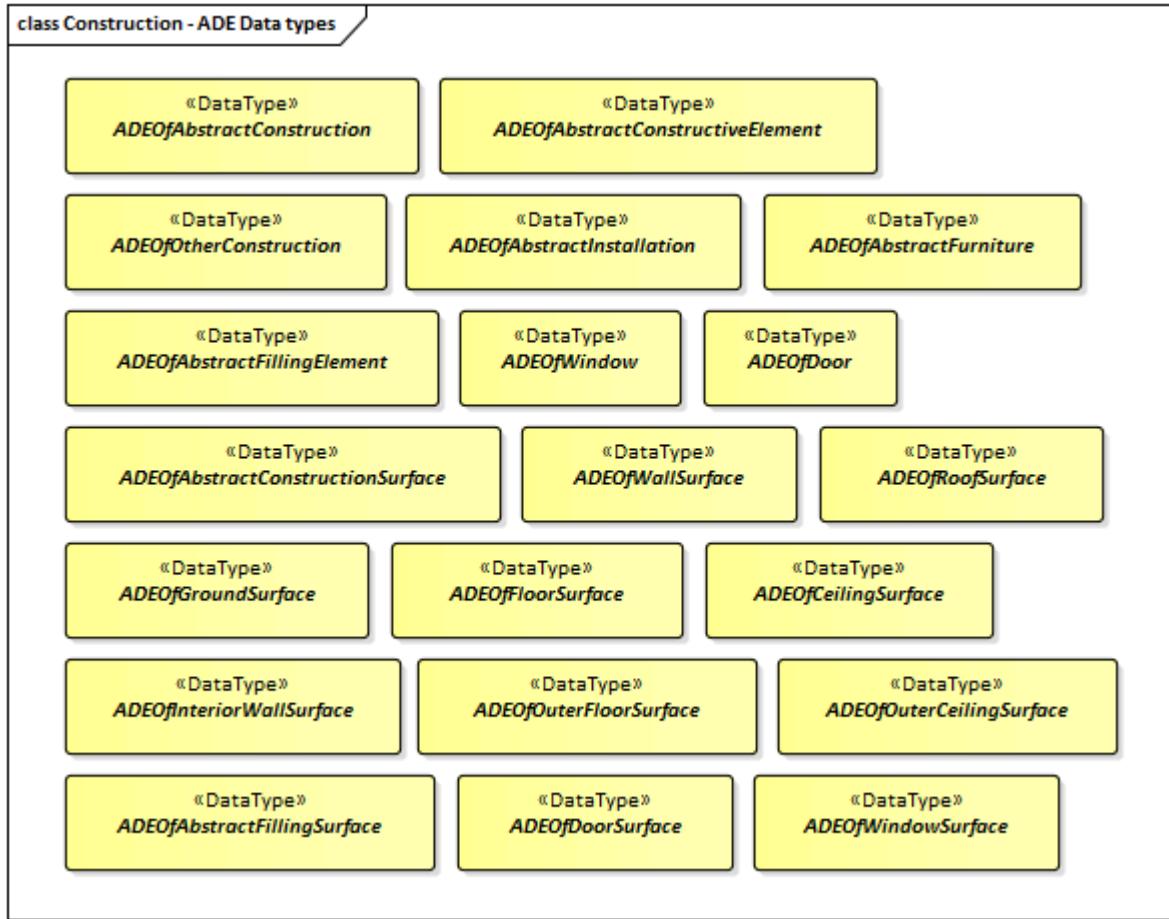


Figure 59 – ADE classes of the CityGML Construction module.

The Code Lists provided for the Construction module are illustrated in Figure 60.

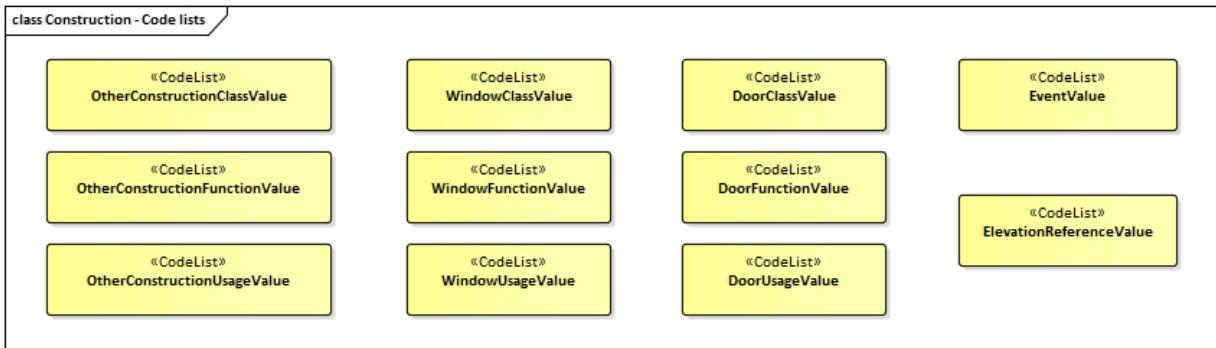


Figure 60 – Codelists from the CityGML Construction module.

Table 56 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Construction module:

Table 56 – Construction space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractConstruction	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
AbstractConstructive Element	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
AbstractFillingElement	No boundaries allowed
AbstractFurniture	No boundaries allowed
AbstractInstallation	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Door	<ul style="list-style-type: none"> • Construction::DoorSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
OtherConstruction	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Window	<ul style="list-style-type: none"> • Construction::WindowSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

7.15.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Construction Module as an Implementation Specification.

REQUIREMENT 36

For each UML class defined or referenced in the Construction Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.

REQUIREMENT 36

F

The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 37

Table 56 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Construction module. An Implementation Specification SHALL only support the boundaries described in Table 56

The use of extension capabilities by Construction elements is constrained by the following requirement:

REQUIREMENT 38

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.15.2. Class definitions

Table 57 – Classes defined in Construction (ApplicationSchema)

NAME	DESCRIPTION
AbstractConstruction «FeatureType»	AbstractConstruction is the abstract superclass for objects that are manufactured by humans from construction materials, are connected to earth, and are intended to be permanent. A connection with the ground also exists when the construction rests by its own weight on the ground or is moveable limited on stationary rails or if the construction is intended to be used mainly stationary.
AbstractConstruction Surface «FeatureType»	AbstractConstructionSurface is the abstract superclass for different kinds of surfaces that bound a construction.
AbstractConstructive Element «FeatureType»	AbstractConstructiveElement is the abstract superclass for the representation of volumetric elements of a construction. Examples are walls, beams, slabs.
AbstractFillingElement «FeatureType»	AbstractFillingElement is the abstract superclass for different kinds of elements that fill the openings of a construction.
AbstractFillingSurface «FeatureType»	AbstractFillingSurface is the abstract superclass for different kinds of surfaces that seal openings filled by filling elements.
AbstractFurniture «Feature Type»	AbstractFurniture is the abstract superclass for the representation of furniture objects of a construction.

NAME	DESCRIPTION
AbstractInstallation «FeatureType»	AbstractInstallation is the abstract superclass for the representation of installation objects of a construction.
CeilingSurface «Feature Type»	A CeilingSurface is a surface that represents the interior ceiling of a construction. An example is the ceiling of a room.
Door «FeatureType»	A Door is a construction for closing an opening intended primarily for access or egress or both. [cf. ISO 6707-1]
DoorSurface «FeatureType»	A DoorSurface is either a boundary surface of a Door feature or a surface that seals an opening filled by a door.
FloorSurface «FeatureType»	A FloorSurface is surface that represents the interior floor of a construction. An example is the floor of a room.
GroundSurface «Feature Type»	A GroundSurface is a surface that represents the ground plate of a construction. The polygon defining the ground plate is congruent with the footprint of the construction.
InteriorWallSurface «FeatureType»	An InteriorWallSurface is a surface that is visible from inside a construction. An example is the wall of a room.
OtherConstruction «Top LevelFeatureType»	An OtherConstruction is a construction that is not covered by any of the other subclasses of AbstractConstruction.
OuterCeilingSurface «FeatureType»	An OuterCeilingSurface is a surface that belongs to the outer building shell with the orientation pointing downwards. An example is the ceiling of a loggia.
OuterFloorSurface «Feature Type»	An OuterFloorSurface is a surface that belongs to the outer construction shell with the orientation pointing upwards. An example is the floor of a loggia.
RoofSurface «FeatureType»	A RoofSurface is a surface that delimits major roof parts of a construction.
WallSurface «FeatureType»	A WallSurface is a surface that is part of the building facade visible from the outside.
Window «FeatureType»	A Window is a construction for closing an opening in a wall or roof, primarily intended to admit light and/or provide ventilation. [cf. ISO 6707-1]
WindowSurface «Feature Type»	A WindowSurface is either a boundary surface of a Window feature or a surface that seals an opening filled by a window.

Table 58 – Data types defined in Construction (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstract Construction «DataType»	ADEOfAbstractConstruction acts as a hook to define properties within an ADE that are to be added to AbstractConstruction.
ADEOfAbstract ConstructionSurface «Data Type»	ADEOfAbstractConstructionSurface acts as a hook to define properties within an ADE that are to be added to AbstractConstructionSurface.
ADEOfAbstract ConstructiveElement «Data Type»	ADEOfAbstractConstructiveElement acts as a hook to define properties within an ADE that are to be added to AbstractConstructiveElement.

NAME	DESCRIPTION
ADEOfAbstractFillingElement «DataType»	ADEOfAbstractFillingElement acts as a hook to define properties within an ADE that are to be added to AbstractFillingElement.
ADEOfAbstractFillingSurface «DataType»	ADEOfAbstractFillingSurface acts as a hook to define properties within an ADE that are to be added to AbstractFillingSurface.
ADEOfAbstractFurniture «DataType»	ADEOfAbstractFurniture acts as a hook to define properties within an ADE that are to be added to AbstractFurniture.
ADEOfAbstractInstallation «DataType»	ADEOfAbstractInstallation acts as a hook to define properties within an ADE that are to be added to AbstractInstallation.
ADEOfCeilingSurface «DataType»	ADEOfCeilingSurface acts as a hook to define properties within an ADE that are to be added to a CeilingSurface.
ADEOfDoor «DataType»	ADEOfDoor acts as a hook to define properties within an ADE that are to be added to a Door.
ADEOfDoorSurface «DataType»	ADEOfDoorSurface acts as a hook to define properties within an ADE that are to be added to a DoorSurface.
ADEOfFloorSurface «DataType»	ADEOfFloorSurface acts as a hook to define properties within an ADE that are to be added to a FloorSurface.
ADEOfGroundSurface «DataType»	ADEOfGroundSurface acts as a hook to define properties within an ADE that are to be added to a GroundSurface.
ADEOfInteriorWallSurface «DataType»	ADEOfInteriorWallSurface acts as a hook to define properties within an ADE that are to be added to an InteriorWallSurface.
ADEOfOtherConstruction «DataType»	ADEOfOtherConstruction acts as a hook to define properties within an ADE that are to be added to an OtherConstruction.
ADEOfOuterCeilingSurface «DataType»	ADEOfOuterCeilingSurface acts as a hook to define properties within an ADE that are to be added to an OuterCeilingSurface.
ADEOfOuterFloorSurface «DataType»	ADEOfOuterFloorSurface acts as a hook to define properties within an ADE that are to be added to an OuterFloorSurface.
ADEOfRoofSurface «DataType»	ADEOfRoofSurface acts as a hook to define properties within an ADE that are to be added to a RoofSurface.
ADEOfWallSurface «DataType»	ADEOfWallSurface acts as a hook to define properties within an ADE that are to be added to a WallSurface.
ADEOfWindow «DataType»	ADEOfWindow acts as a hook to define properties within an ADE that are to be added to a Window.
ADEOfWindowSurface «DataType»	ADEOfWindowSurface acts as a hook to define properties within an ADE that are to be added to a WindowSurface.
ConstructionEvent «DataType»	A ConstructionEvent is a data type used to describe a specific event that is associated with a construction. Examples are the issuing of a building permit or the renovation of a building.
Elevation «DataType»	Elevation is a data type that includes the elevation value itself and information on how this elevation was measured. [cf. INSPIRE]

NAME	DESCRIPTION
Height «DataType»	Height represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Table 59 – Enumerated classes defined in Construction (ApplicationSchema)

NAME	DESCRIPTION
ConditionOfConstructionValue «Enumeration»	ConditionOfConstructionValue enumerates different conditions of a construction. [cf. INSPIRE]
HeightStatusValue «Enumeration»	HeightStatusValue enumerates the different methods used to capture a height. [cf. INSPIRE]
RelationToConstruction «Enumeration»	RelationToConstruction is an enumeration used to describe whether an installation is positioned inside and/or outside of a construction.

Table 60 – Code list classes defined in Construction (ApplicationSchema)

NAME	DESCRIPTION
DoorClassValue «CodeList»	DoorClassValue is a code list used to further classify a Door.
DoorFunctionValue «CodeList»	DoorFunctionValue is a code list that enumerates the different purposes of a Door.
DoorUsageValue «CodeList»	DoorUsageValue is a code list that enumerates the different uses of a Door.
ElevationReferenceValue «CodeList»	ElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure construction heights.
EventValue «CodeList»	EventValue is a code list that enumerates the different events of a construction.
OtherConstructionClassValue «CodeList»	OtherConstructionClassValue is a code list used to further classify an Other Construction.
OtherConstructionFunctionValue «CodeList»	OtherConstructionFunctionValue is a code list that enumerates the different purposes of an OtherConstruction.
OtherConstructionUsageValue «CodeList»	OtherConstructionUsageValue is a code list that enumerates the different uses of an OtherConstruction.
WindowClassValue «CodeList»	WindowClassValue is a code list used to further classify a Window.
WindowFunctionValue «CodeList»	WindowFunctionValue is a code list that enumerates the different purposes of a Window.
WindowUsageValue «CodeList»	WindowUsageValue is a code list that enumerates the different uses of a Window.

7.15.3. Additional Information

Additional information about the Construction Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.16. Bridge

REQUIREMENTS CLASS 15

Target type	Implementation Specification
Dependency	/req/req-class-core
Dependency	/req/req-class-construction

The Bridge module provides the representation of thematic and spatial aspects of bridges. Bridges are movable or unmovable structures that span intervening natural or built elements. In this way, bridges allow the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. Bridges are represented in the UML model by the top-level feature type *Bridge*, which is the main class of the Bridge module. Bridges can physically or functionally be subdivided into bridge parts. In addition, bridges can be decomposed into structural elements, such as pylons, anchorages, cables, slabs, and beams.

The free space inside bridges is represented by rooms, which allows a virtual accessibility of bridges. Bridges can contain installations and furniture. Installations are permanent parts of a bridge that strongly affect the outer or inner appearance of the bridge and that cannot be moved. Examples are stairways, signals, railings, and lamps. Furniture, in contrast, represent moveable objects of a bridge, like signs, art works, and benches. Bridges can be bounded by different types of surfaces. In this way, the outer structure of bridges can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of rooms can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of bridges, i.e., windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the Bridge module is depicted in Figure 61. The Bridge module inherits concepts from the Construction module (cf. Clause 7.15). The Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of Requirements Class Bridge can be found in the [CityGML 3.0 Users Guide](#).

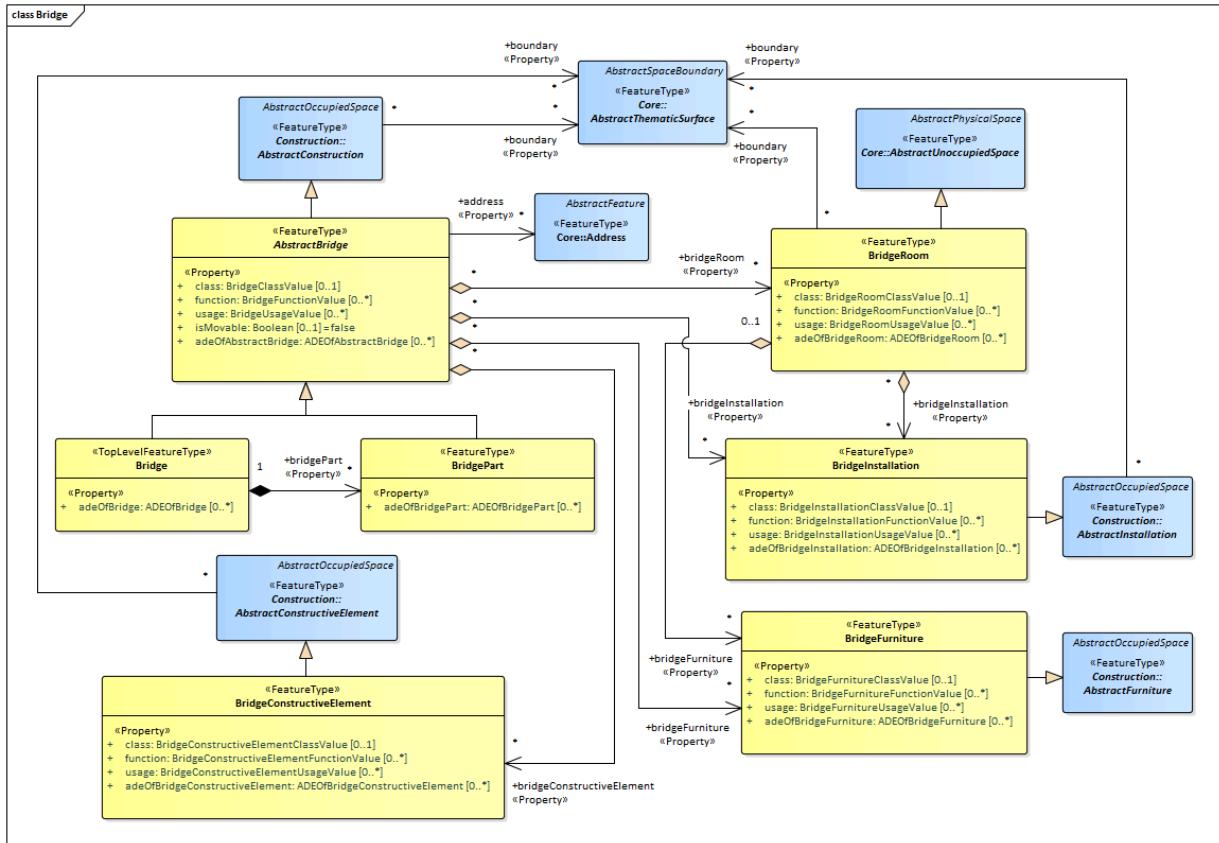


Figure 61 – UML diagram of the Bridge Model.

The ADE data types provided for the Bridge module are illustrated in Figure 62.

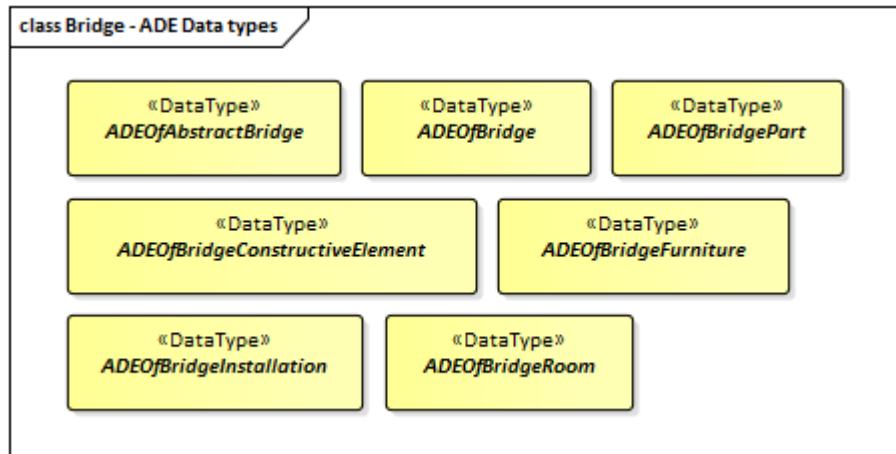


Figure 62 – ADE classes of the CityGML Bridge module.

The Code Lists provided for the Bridge module are illustrated in Figure 63.

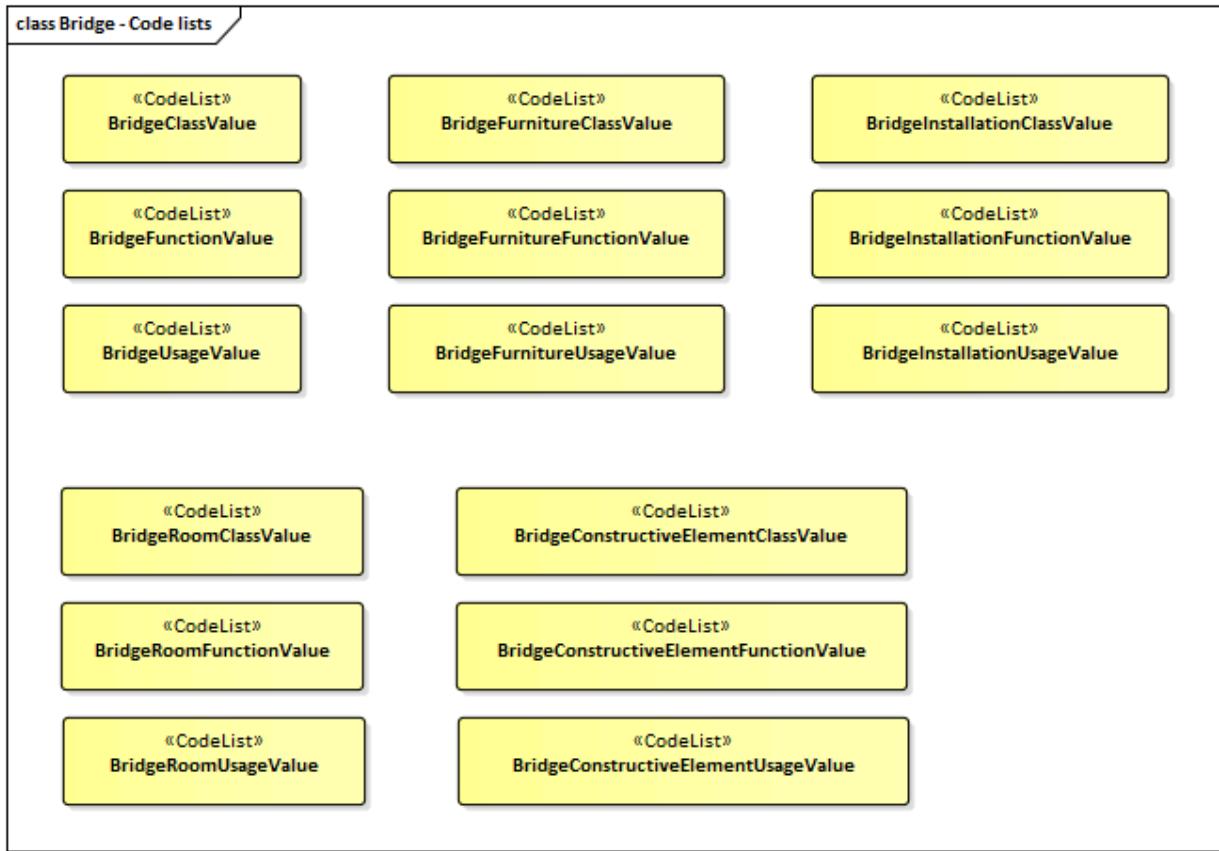


Figure 63 – Codelists from the CityGML Bridge module.

Table 61 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Bridge module:

Table 61 – Bridge space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractBridge	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
Bridge	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BridgeConstructiveElement	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BridgeFurniture	No boundaries allowed
BridgeInstallation	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BridgePart	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface

SPACE CLASS	ALLOWED SPACE BOUNDARIES
BridgeRoom	<ul style="list-style-type: none"> • Generics::GenericThematicSurface • possible classes from ADEs • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

7.16.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Bridge Module as an Implementation Specification.

REQUIREMENT 39

For each UML class defined or referenced in the Bridge Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 40

Table 61 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Bridge module. An Implementation Specification SHALL only support the boundaries described in Table 61

The use of extension capabilities by Bridge elements is constrained by the following requirement:

REQUIREMENT 41

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.16.2. Class definitions

Table 62 – Classes defined in Bridge (ApplicationSchema)

NAME	DESCRIPTION
AbstractBridge «Feature Type»	AbstractBridge is an abstract superclass representing the common attributes and associations of the classes Bridge and BridgePart.
Bridge «TopLevelFeature Type»	A Bridge represents a structure that affords the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. [cf. ISO 6707-1]
BridgeConstructiveElement «FeatureType»	A BridgeConstructiveElement is an element of a bridge which is essential from a structural point of view. Examples are pylons, anchorages, slabs, beams.
BridgeFurniture «Feature Type»	A BridgeFurniture is an equipment for occupant use, usually not fixed to the bridge. [cf. ISO 6707-1]
BridgeInstallation «Feature Type»	A BridgelInstallation is a permanent part of a Bridge (inside and/or outside) which does not have the significance of a BridgePart. In contrast to BridgeConstructiveElements, a BridgelInstallation is not essential from a structural point of view. Examples are stairs, antennas or railways.
BridgePart «FeatureType»	A BridgePart is a physical or functional subdivision of a Bridge. It would be considered a Bridge, if it were not part of a collection of other BridgeParts.
BridgeRoom «FeatureType»	A BridgeRoom is a space within a Bridge or BridgePart intended for human occupancy (e.g. a place of work or recreation) and/or containment (storage) of animals or things. A BridgeRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or Generic Surfaces).

Table 63 – Data types defined in Bridge (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractBridge «DataType»	ADEOfAbstractBridge acts as a hook to define properties within an ADE that are to be added to AbstractBridge.
ADEOfBridge «DataType»	ADEOfBridge acts as a hook to define properties within an ADE that are to be added to a Bridge.
ADEOfBridgeConstructiveElement «DataType»	ADEOfBridgeConstructiveElement acts as a hook to define properties within an ADE that are to be added to a BridgeConstructiveElement.
ADEOfBridgeFurniture «DataType»	ADEOfBridgeFurniture acts as a hook to define properties within an ADE that are to be added to a BridgeFurniture.
ADEOfBridgeInstallation «DataType»	ADEOfBridgeInstallation acts as a hook to define properties within an ADE that are to be added to a BridgeInstallation.
ADEOfBridgePart «DataType»	ADEOfBridgePart acts as a hook to define properties within an ADE that are to be added to a BridgePart.
ADEOfBridgeRoom «DataType»	ADEOfBridgeRoom acts as a hook to define properties within an ADE that are to be added to a BridgeRoom.

Table 64 – Code list classes defined in Bridge (ApplicationSchema)

NAME	DESCRIPTION
BridgeClassValue «CodeList»	BridgeClassValue is a code list used to further classify a Bridge.
BridgeConstructiveElementClassValue «CodeList»	BridgeConstructiveElementClassValue is a code list used to further classify a BridgeConstructiveElement.
BridgeConstructiveElementFunctionValue «CodeList»	BridgeConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BridgeConstructiveElement.
BridgeConstructiveElementUsageValue «CodeList»	BridgeConstructiveElementUsageValue is a code list that enumerates the different uses of a BridgeConstructiveElement.
BridgeFunctionValue «CodeList»	BridgeFunctionValue is a code list that enumerates the different purposes of a Bridge.
BridgeFurnitureClassValue «CodeList»	BridgeFurnitureClassValue is a code list used to further classify a BridgeFurniture.
BridgeFurnitureFunctionValue «CodeList»	BridgeFurnitureFunctionValue is a code list that enumerates the different purposes of a BridgeFurniture.
BridgeFurnitureUsageValue «CodeList»	BridgeFurnitureUsageValue is a code list that enumerates the different uses of a BridgeFurniture.
BridgeInstallationClassValue «CodeList»	BridgeInstallationClassValue is a code list used to further classify a BridgeInstallation.
BridgeInstallationFunctionValue «CodeList»	BridgeInstallationFunctionValue is a code list that enumerates the different purposes of a BridgeInstallation.

NAME	DESCRIPTION
BridgeInstallationUsageValue «CodeList»	BridgeInstallationUsageValue is a code list that enumerates the different uses of a BridgeInstallation.
BridgeRoomClassValue «CodeList»	BridgeRoomClassValue is a code list used to further classify a BridgeRoom.
BridgeRoomFunctionValue «CodeList»	BridgeRoomFunctionValue is a code list that enumerates the different purposes of a BridgeRoom.
BridgeRoomUsageValue «CodeList»	BridgeRoomUsageValue is a code list that enumerates the different uses of a BridgeRoom.
BridgeUsageValue «CodeList»	BridgeUsageValue is a code list that enumerates the different uses of a Bridge.

7.16.3. Additional Information

Additional information about the Bridge Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.17. Building

REQUIREMENTS CLASS 16

Target type	Implementation Specification
Dependency	/req/req-class-core
Dependency	/req/req-class-construction

The Building module provides the representation of thematic and spatial aspects of buildings. Buildings are free-standing, self-supporting constructions that are roofed and usually walled, and that can be entered by humans and are normally designed to stand permanently in one place. Buildings are intended for human occupancy (e.g., a place of work or recreation), habitation and/or shelter of humans, animals or things. Buildings are represented in the UML model by the top-level feature type *Building*, which is the main class of the Building module. Buildings can physically or functionally be subdivided into building parts, and logically into storeys and building units (e.g., apartments). In addition, buildings can be decomposed into structural elements, such as walls, slabs, staircases, and beams.

The interior of buildings is represented by rooms. This allows a virtual accessibility of buildings such as for visitor information in a museum (“Location Based Services”), the examination of accommodation standards or the presentation of daylight illumination of a building.

Buildings can contain installations and furniture. Installations are permanent parts of a building that strongly affect the outer or inner appearance of the building and that cannot be moved. Examples are balconies, chimneys, dormers or stairs. In contrast, furniture, represents moveable objects inside a building, like tables and chairs.

Buildings can be bounded by different types of surfaces. In this way, the outer façade of buildings can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of rooms can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of buildings, i.e. windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the building module is depicted in Figure 64. The Building module inherits concepts from the Construction module (cf. Clause 7.15). The Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of the Requirements Class Building can be found in the [CityGML 3.0 Users Guide](#).

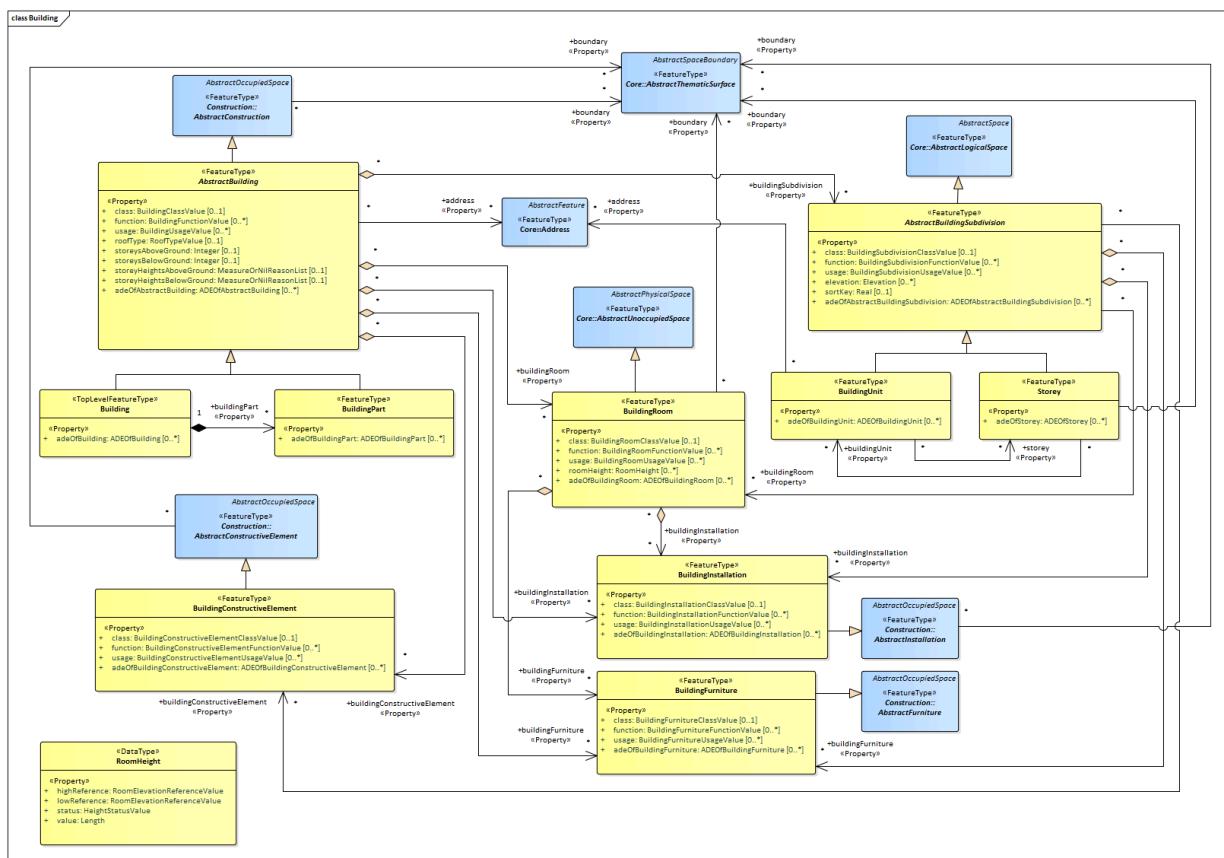


Figure 64 – UML diagram of CityGML's building model.

The ADE data types provided for the Building module are illustrated in Figure 65.

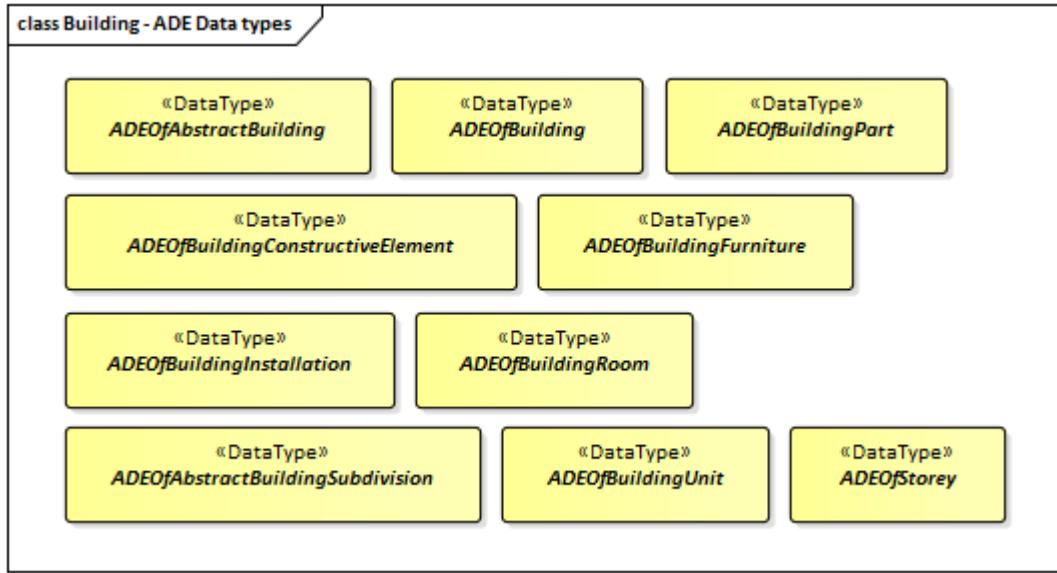


Figure 65 – ADE classes of the CityGML Building module.

The Code Lists provided for the Building module are illustrated in Figure 66.

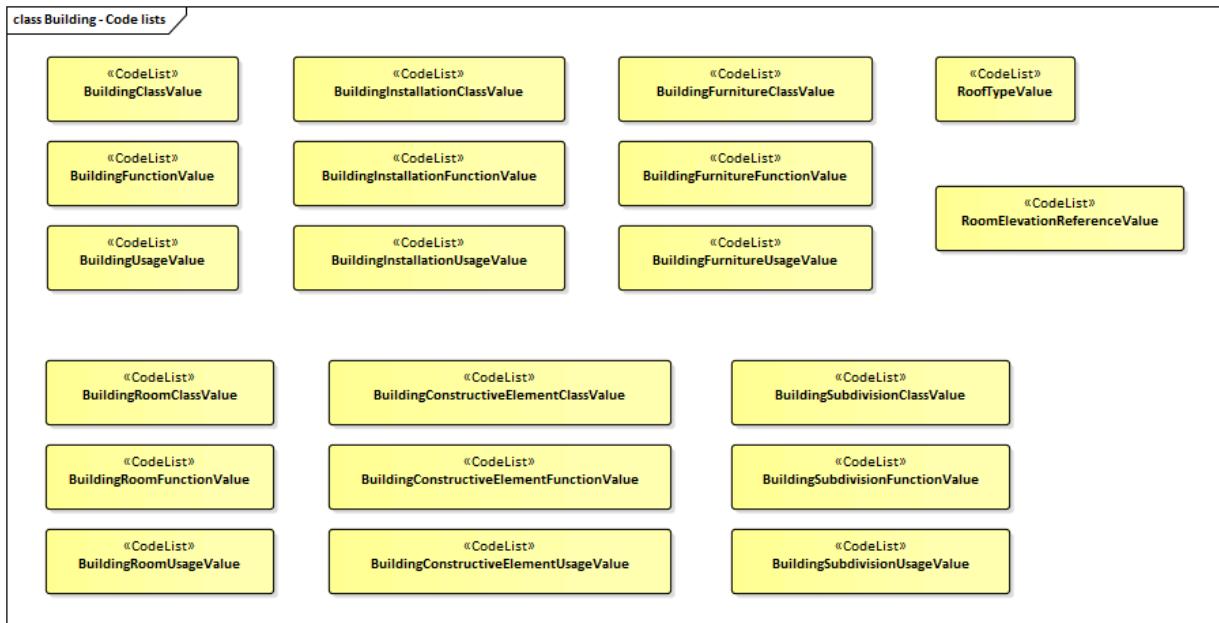


Figure 66 – Codelists from the CityGML Building module.

Table 65 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Building module:

Table 65 – Building space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractBuilding	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
AbstractBuildingSubdivision	No boundaries allowed
Building	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BuildingConstructive Element	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BuildingFurniture	No boundaries allowed

SPACE CLASS	ALLOWED SPACE BOUNDARIES
BuildingInstallation	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BuildingPart	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BuildingRoom	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
BuildingUnit	<ul style="list-style-type: none"> • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Storey	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface

SPACE CLASS	ALLOWED SPACE BOUNDARIES
	<ul style="list-style-type: none"> Core::ClosureSurface Generics::GenericThematicSurface possible classes from ADEs

7.17.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Building Module as an Implementation Specification.

REQUIREMENT 42

For each UML class defined or referenced in the Building Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 43

Table 65 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Building module. An Implementation Specification SHALL only support the boundaries described in Table 65

The use of extension capabilities by Building elements is constrained by the following requirement:

REQUIREMENT 44

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.17.2. Class definitions

Table 66 – Classes defined in Building (ApplicationSchema)

NAME	DESCRIPTION
AbstractBuilding «Feature Type»	AbstractBuilding is an abstract superclass representing the common attributes and associations of the classes Building and BuildingPart.
AbstractBuildingSubdivision «FeatureType»	AbstractBuildingSubdivision is the abstract superclass for different kinds of logical building subdivisions.
Building «TopLevelFeature Type»	A Building is a free-standing, self-supporting construction that is roofed, usually walled, and can be entered by humans and is normally designed to stand permanently in one place. It is intended for human occupancy (e.g. a place of work or recreation), habitation and/or shelter of humans, animals or things.
BuildingConstructive Element «FeatureType»	A BuildingConstructiveElement is an element of a Building which is essential from a structural point of view. Examples are walls, slabs, staircases, beams.
BuildingFurniture «Feature Type»	A BuildingFurniture is an equipment for occupant use, usually not fixed to the building. [cf. ISO 6707-1]
BuildingInstallation «FeatureType»	A BuildingInstallation is a permanent part of a Building (inside and/or outside) which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs.
BuildingPart «FeatureType»	A BuildingPart is a physical or functional subdivision of a Building. It would be considered a Building, if it were not part of a collection of other BuildingParts.
BuildingRoom «Feature Type»	A BuildingRoom is a space within a Building or BuildingPart intended for human occupancy (e.g. a place of work or recreation) and/or containment of animals or things. A BuildingRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
BuildingUnit «FeatureType»	A BuildingUnit is a logical subdivision of a Building. BuildingUnits are formed according to some homogeneous property like function, ownership, management, or accessibility. They may be separately sold, rented out, inherited, managed, etc.
Storey «FeatureType»	A Storey is typically a horizontal section of a Building. Storeys are not always defined according to the building structure, but can also be defined according to logical considerations.

Table 67 – Data types defined in Building (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractBuilding «DataType»	ADEOfAbstractBuilding acts as a hook to define properties within an ADE that are to be added to AbstractBuilding.
ADEOfAbstractBuildingSubdivision «DataType»	ADEOfAbstractBuildingSubdivision acts as a hook to define properties within an ADE that are to be added to AbstractBuildingSubdivision.
ADEOfBuilding «DataType»	ADEOfBuilding acts as a hook to define properties within an ADE that are to be added to a Building.
ADEOfBuildingConstructiveElement «DataType»	ADEOfBuildingConstructiveElement acts as a hook to define properties within an ADE that are to be added to a BuildingConstructiveElement.
ADEOfBuildingFurniture «DataType»	ADEOfBuildingFurniture acts as a hook to define properties within an ADE that are to be added to a BuildingFurniture.
ADEOfBuildingInstallation «DataType»	ADEOfBuildingInstallation acts as a hook to define properties within an ADE that are to be added to a BuildingInstallation.
ADEOfBuildingPart «DataType»	ADEOfBuildingPart acts as a hook to define properties within an ADE that are to be added to a BuildingPart.
ADEOfBuildingRoom «DataType»	ADEOfBuildingRoom acts as a hook to define properties within an ADE that are to be added to a BuildingRoom.
ADEOfBuildingUnit «DataType»	ADEOfBuildingUnit acts as a hook to define properties within an ADE that are to be added to a BuildingUnit.
ADEOfStorey «DataType»	ADEOfStorey acts as a hook to define properties within an ADE that are to be added to a Storey.
RoomHeight «DataType»	The RoomHeight represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]

Table 68 – Code list classes defined in Building (ApplicationSchema)

NAME	DESCRIPTION
BuildingClassValue «CodeList»	BuildingClassValue is a code list used to further classify a Building.
BuildingConstructiveElementClassValue «CodeList»	BuildingConstructiveElementClassValue is a code list used to further classify a BuildingConstructiveElement.
BuildingConstructiveElementFunctionValue «CodeList»	BuildingConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BuildingConstructiveElement.
BuildingConstructiveElementUsageValue «CodeList»	BuildingConstructiveElementUsageValue is a code list that enumerates the different uses of a BuildingConstructiveElement.

NAME	DESCRIPTION
BuildingFunctionValue «CodeList»	BuildingFunctionValue is a code list that enumerates the different purposes of a Building.
BuildingFurnitureClassValue «CodeList»	BuildingFurnitureClassValue is a code list used to further classify a BuildingFurniture.
BuildingFurnitureFunctionValue «CodeList»	BuildingFurnitureFunctionValue is a code list that enumerates the different purposes of a BuildingFurniture.
BuildingFurnitureUsageValue «CodeList»	BuildingFurnitureUsageValue is a code list that enumerates the different uses of a BuildingFurniture.
BuildingInstallationClassValue «CodeList»	BuildingInstallationClassValue is a code list used to further classify a Building Installation.
BuildingInstallationFunctionValue «CodeList»	BuildingInstallationFunctionValue is a code list that enumerates the different purposes of a BuildingInstallation.
BuildingInstallationUsageValue «CodeList»	BuildingInstallationUsageValue is a code list that enumerates the different uses of a BuildingInstallation.
BuildingRoomClassValue «CodeList»	BuildingRoomClassValue is a code list used to further classify a BuildingRoom.
BuildingRoomFunctionValue «CodeList»	BuildingRoomFunctionValue is a code list that enumerates the different purposes of a BuildingRoom.
BuildingRoomUsageValue «CodeList»	BuildingRoomUsageValue is a code list that enumerates the different uses of a Building Room.
BuildingSubdivisionClassValue «CodeList»	BuildingSubdivisionClassValue is a code list used to further classify a Building Subdivision.
BuildingSubdivisionFunctionValue «CodeList»	BuildingSubdivisionFunctionValue is a code list that enumerates the different purposes of a BuildingSubdivision.
BuildingSubdivisionUsageValue «CodeList»	BuildingSubdivisionUsageValue is a code list that enumerates the different uses of a BuildingSubdivision.
BuildingUsageValue «CodeList»	BuildingUsageValue is a code list that enumerates the different uses of a Building.
RoofTypeValue «CodeList»	RoofTypeValue is a code list that enumerates different roof types.
RoomElevationReferenceValue «CodeList»	RoomElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure room heights.

7.17.3. Additional Information

Additional information about the Building Module can be found in the [OGC CityGML 3.0 Users Guide](#).

7.18. Tunnel

REQUIREMENTS CLASS 17

Target type	Implementation Specification
Dependency	/req/req-class-core
Dependency	/req/req-class-construction

The Tunnel module provides the representation of thematic and spatial aspects of tunnels. Tunnels are horizontal or sloping enclosed passage ways of a certain length, mainly underground or underwater. Tunnels are intended for passing obstacles such as mountains, waterways, or other traffic routes by humans, animals, or things.

Tunnels are represented in the UML model by the top-level feature type *Tunnel*, which is the main class of the Tunnel module. Tunnels can physically or functionally be subdivided into tunnel parts. In addition, tunnels can be decomposed into structural elements, such as walls, slabs, staircases, and beams.

The interior of tunnels is represented by hollow spaces. This allows a virtual accessibility of tunnels such as for driving through a tunnel, for simulating disaster management, or for presenting the light illumination within a tunnel.

Tunnels can contain installations and furniture. Installations are permanent parts of a tunnel that strongly affect the outer or inner appearance of the tunnel and that cannot be moved. Examples are stairs, railings, radiators, or pipes. In contrast, furniture represents moveable objects inside a tunnel, like movable equipment in control areas.

Tunnels can be bounded by different types of surfaces. In this way, the outer structure of tunnels can be differentiated semantically into wall surfaces, roof surfaces, ground surfaces, outer floor surfaces, and outer ceiling surfaces, whereas the visible surface of hollow spaces can be structured into interior wall surfaces, floor surfaces, and ceiling surfaces. Furthermore, the openings of tunnels, i.e., windows and doors, can be represented including their corresponding surfaces.

The UML diagram of the Tunnel module is depicted in Figure 67. The Tunnel module inherits concepts from the Construction module (cf. Clause 7.15). The Construction module defines objects that are common to all types of construction, such as the different surface types and the openings. A detailed discussion of the Requirements Class Tunnel can be found in the [CityGML 3.0 Users Guide](#).

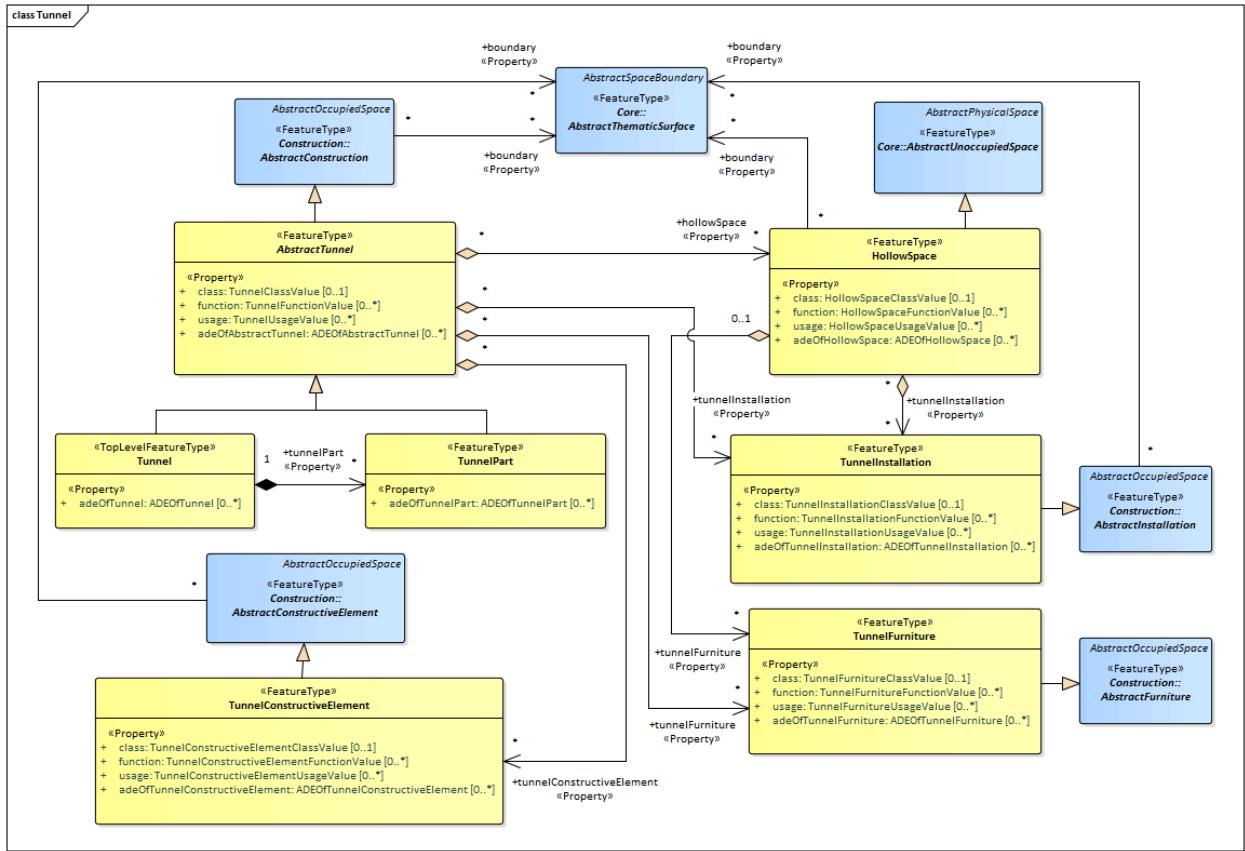


Figure 67 – UML diagram of the Tunnel Model.

The ADE data types provided for the Tunnel module are illustrated in Figure 68.

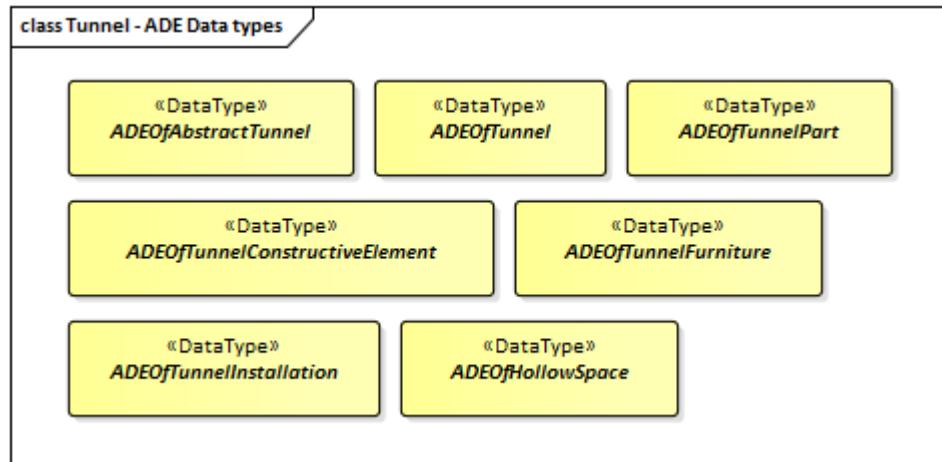


Figure 68 – ADE classes of the CityGML Tunnel module.

The Code Lists provided for the Tunnel module are illustrated in Figure 69.

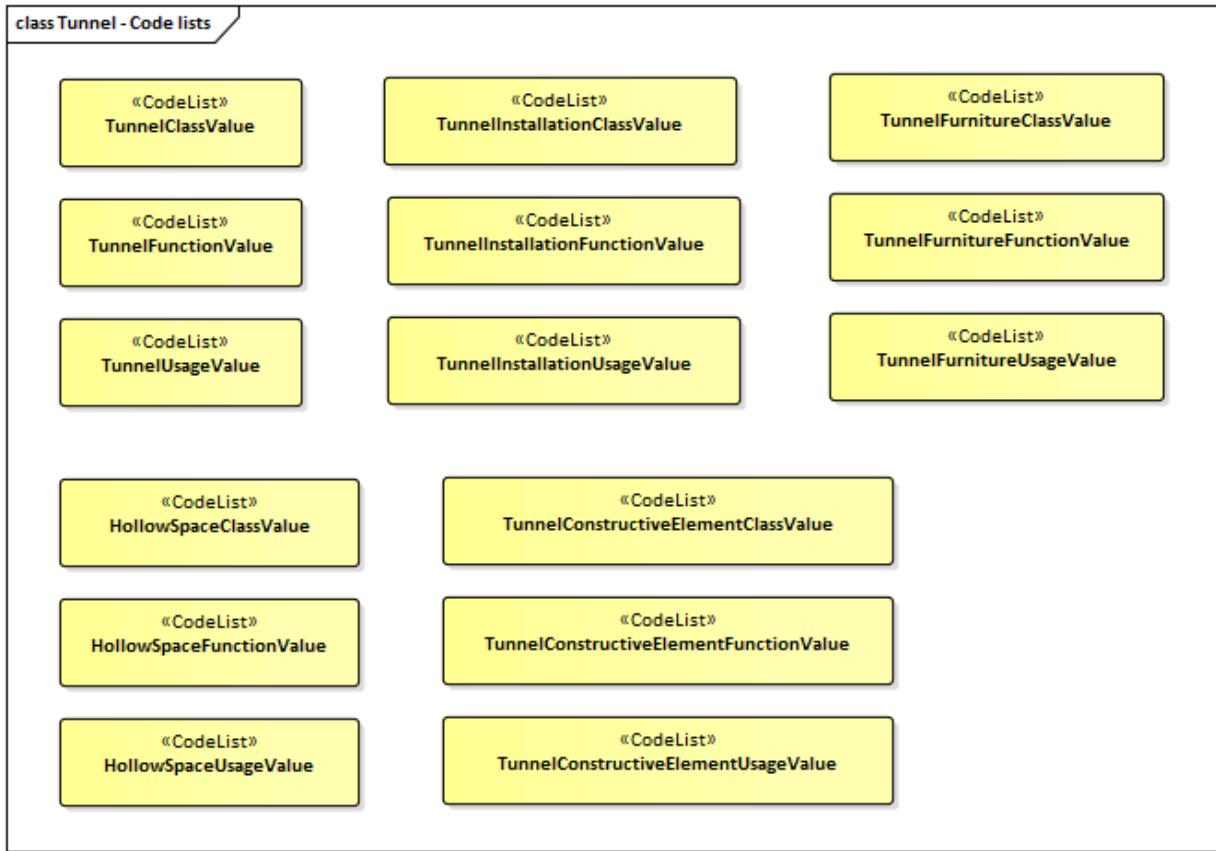


Figure 69 – Codelists from the CityGML Tunnel module.

Table 69 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Tunnel module:

Table 69 – Tunnel space classes and their allowed thematic surface boundaries

SPACE CLASS	ALLOWED SPACE BOUNDARIES
AbstractTunnel	<ul style="list-style-type: none">• Construction::AbstractConstructionSurface and all subclasses, i.e.<ul style="list-style-type: none">• Construction::GroundSurface,• Construction::RoofSurface,• Construction::CeilingSurface,• Construction::OuterCeilingSurface,• Construction::FloorSurface,• Construction::OuterFloorSurface,• Construction::WallSurface,• Construction::InteriorWallSurface• Core::ClosureSurface• Generics::GenericThematicSurface• possible classes from ADEs

SPACE CLASS	ALLOWED SPACE BOUNDARIES
HollowSpace	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
Tunnel	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
TunnelConstructiveElement	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs
TunnelFurniture	No boundaries allowed
TunnelInstallation	<ul style="list-style-type: none"> • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface

SPACE CLASS	ALLOWED SPACE BOUNDARIES
TunnelPart	<ul style="list-style-type: none"> • Generics::GenericThematicSurface • possible classes from ADEs • Construction::AbstractConstructionSurface and all subclasses, i.e. <ul style="list-style-type: none"> • Construction::GroundSurface, • Construction::RoofSurface, • Construction::CeilingSurface, • Construction::OuterCeilingSurface, • Construction::FloorSurface, • Construction::OuterFloorSurface, • Construction::WallSurface, • Construction::InteriorWallSurface • Core::ClosureSurface • Generics::GenericThematicSurface • possible classes from ADEs

7.18.1. Requirements

The following requirement defines the rules governing implementation of the CityGML Tunnel Module as an Implementation Specification.

REQUIREMENT 45

For each UML class defined or referenced in the Tunnel Package:

A	The Implementation Specification SHALL contain an element which represents the same concept as that defined for the UML class.
B	The Implementation Specification SHALL represent associations with the same source, target, direction, roles, and multiplicities as those of the UML class.
C	The implementation Specification SHALL represent the attributes of the UML class including the name, definition, type, and multiplicity.
D	The implementation Specification SHALL represent the attributes of all superclasses of the UML class including the name, definition, type, and multiplicity.
E	The implementation Specification SHALL represent the associations of all superclasses of the UML class including the source, target, direction, roles, and multiplicity.
F	The Implementation Specification SHALL specify how an implementation observes all constraints the Conceptual Model imposes on the UML class.

The implementation of this Module is further constrained by the following spatial boundary requirement:

REQUIREMENT 46

Table 69 lists the surfaces that are allowed as thematic surface boundaries of the space classes defined in the Tunnel module. An Implementation Specification SHALL only support the boundaries described in Table 69

The use of extension capabilities by Tunnel elements is constrained by the following requirement:

REQUIREMENT 47

ADE element and property extensions SHALL NOT be used unless conformance with the ADE Requirements Class can be demonstrated.

7.18.2. Class definitions

Table 70 – Classes defined in Tunnel (ApplicationSchema)

NAME	DESCRIPTION
AbstractTunnel «Feature Type»	AbstractTunnel is an abstract superclass representing the common attributes and associations of the classes Tunnel and TunnelPart.
HollowSpace «FeatureType»	A HollowSpace is a space within a Tunnel or TunnelPart intended for certain functions (e.g. transport or passage ways, service rooms, emergency shelters). A HollowSpace is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
Tunnel «TopLevelFeature Type»	A Tunnel represents a horizontal or sloping enclosed passage way of a certain length, mainly underground or underwater. [cf. ISO 6707-1]
TunnelConstructiveElement «FeatureType»	A TunnelConstructiveElement is an element of a Tunnel which is essential from a structural point of view. Examples are walls, slabs, beams.
TunnelFurniture «Feature Type»	A TunnelFurniture is an equipment for occupant use, usually not fixed to the tunnel. [cf. ISO 6707-1]
TunnellInstallation «Feature Type»	A TunnellInstallation is a permanent part of a Tunnel (inside and/or outside) which does not have the significance of a TunnelPart. In contrast to TunnelConstructiveElement, a TunnellInstallation is not essential from a structural point of view. Examples are stairs, antennas or railings.
TunnelPart «FeatureType»	A TunnelPart is a physical or functional subdivision of a Tunnel. It would be considered a Tunnel, if it were not part of a collection of other TunnelParts.

Table 71 – Data types defined in Tunnel (ApplicationSchema)

NAME	DESCRIPTION
ADEOfAbstractTunnel «DataType»	ADEOfAbstractTunnel acts as a hook to define properties within an ADE that are to be added to AbstractTunnel.
ADEOfHollowSpace «DataType»	ADEOfHollowSpace acts as a hook to define properties within an ADE that are to be added to a HollowSpace.
ADEOfTunnel «DataType»	ADEOfTunnel acts as a hook to define properties within an ADE that are to be added to a Tunnel.
ADEOfTunnelConstructiveElement «DataType»	ADEOfTunnelConstructiveElement acts as a hook to define properties within an ADE that are to be added to a TunnelConstructiveElement.
ADEOfTunnelFurniture «DataType»	ADEOfTunnelFurniture acts as a hook to define properties within an ADE that are to be added to a TunnelFurniture.
ADEOfTunnelInstallation «DataType»	ADEOfTunnelInstallation acts as a hook to define properties within an ADE that are to be added to a TunnelInstallation.
ADEOfTunnelPart «DataType»	ADEOfTunnelPart acts as a hook to define properties within an ADE that are to be added to a TunnelPart.

Table 72 – Code list classes defined in Tunnel (ApplicationSchema)

NAME	DESCRIPTION
HollowSpaceClassValue «CodeList»	HollowSpaceClassValue is a code list used to further classify a HollowSpace.
HollowSpaceFunctionValue «CodeList»	HollowSpaceFunctionValue is a code list that enumerates the different purposes of a HollowSpace.
HollowSpaceUsageValue «CodeList»	HollowSpaceUsageValue is a code list that enumerates the different uses of a HollowSpace.
TunnelClassValue «CodeList»	TunnelClassValue is a code list used to further classify a Tunnel.
TunnelConstructiveElementClassValue «CodeList»	TunnelConstructiveElementClassValue is a code list used to further classify a TunnelConstructiveElement.
TunnelConstructiveElementFunctionValue «CodeList»	TunnelConstructiveElementFunctionValue is a code list that enumerates the different purposes of a TunnelConstructiveElement.
TunnelConstructiveElementUsageValue «CodeList»	TunnelConstructiveElementUsageValue is a code list that enumerates the different uses of a TunnelConstructiveElement.
TunnelFunctionValue «CodeList»	TunnelFunctionValue is a code list that enumerates the different purposes of a Tunnel.
TunnelFurnitureClassValue «CodeList»	TunnelFurnitureClassValue is a code list used to further classify a TunnelFurniture.
TunnelFurnitureFunctionValue «CodeList»	TunnelFurnitureFunctionValue is a code list that enumerates the different purposes of a TunnelFurniture.

NAME	DESCRIPTION
TunnelFurnitureUsageValue «CodeList»	TunnelFurnitureUsageValue is a code list that enumerates the different uses of a TunnelFurniture.
TunnellInstallationClassValue «CodeList»	TunnellInstallationClassValue is a code list used to further classify a TunnellInstallation.
TunnellInstallationFunctionValue «CodeList»	TunnellInstallationFunctionValue is a code list that enumerates the different purposes of a TunnellInstallation.
TunnellInstallationUsageValue «CodeList»	TunnellInstallationUsageValue is a code list that enumerates the different uses of a TunnellInstallation.
TunnelUsageValue «CodeList»	TunnelUsageValue is a code list that enumerates the different uses of a Tunnel.

7.18.3. Additional Information

Additional information about the Tunnel Module can be found in the [OGC CityGML 3.0 Users Guide](#).

8

CITYGML DATA DICTIONARY

CITYGML DATA DICTIONARY

The CityGML UML model is the normative definition of the CityGML Conceptual Model. The Data Dictionary tables in this section were software generated from the UML model. As such, this section provides a normative representation of the CityGML Conceptual Model.

An alternate representation can be found in the Conceptual Model in Clause 7.

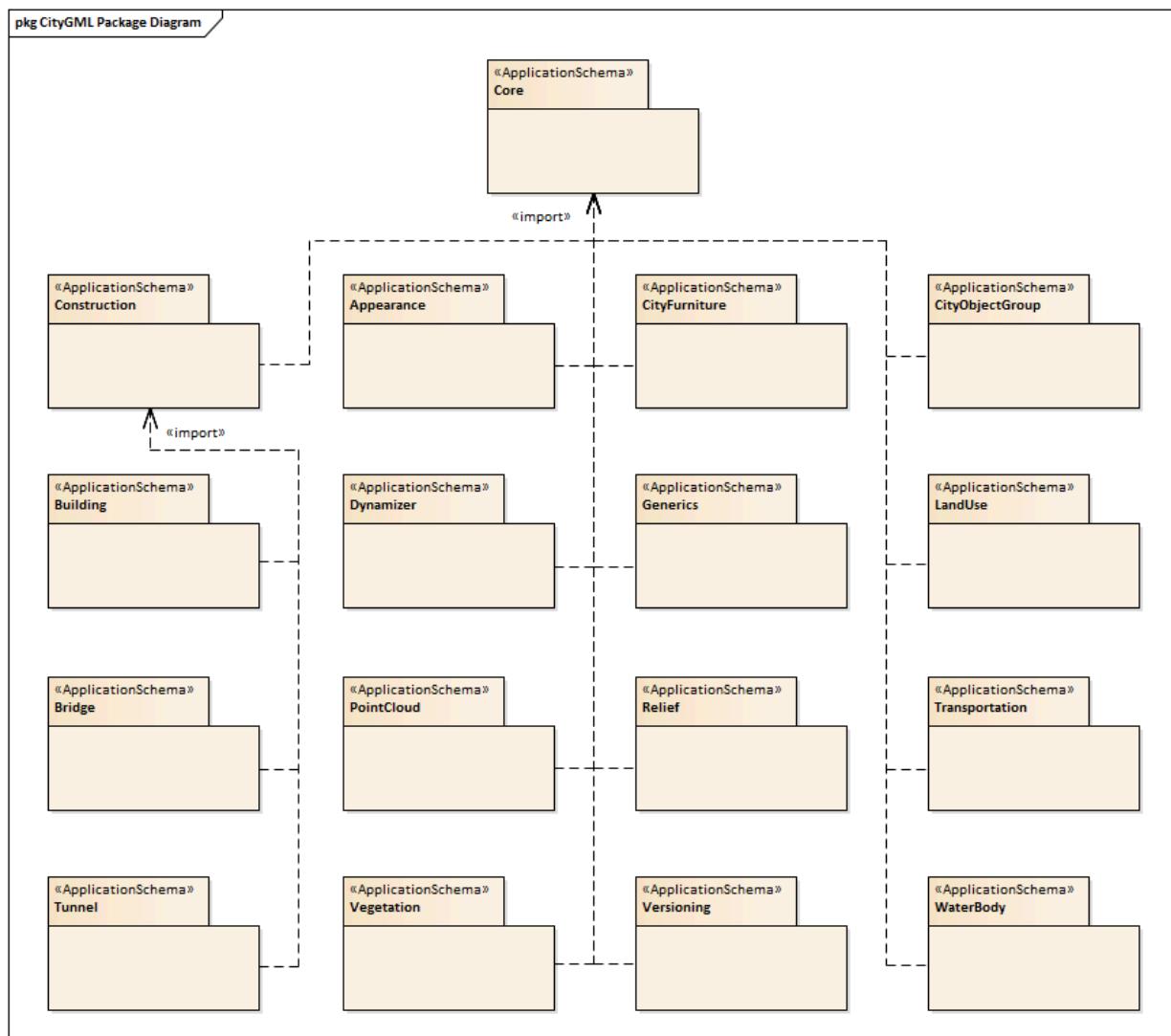


Figure 70 – CityGML UML Packages

8.1. ISO Classes

The following classes are defined in ISO standards and used by the CityGML Conceptual Model.

8.1.1. AnyFeature (from ISO 19109:2015)

Table 73 – Metadata of AnyFeature (class)

DEFINITION:	AnyFeature is an abstract class that is the generalization of all feature types. AnyFeature is an instance of the «metaclass» FeatureType [cf. ISO 19109].
SUBCLASS OF:	none
STEREOTYPE:	«FeatureType»

Table 74 – Associations of AnyFeature (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
	FeatureType [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.2. CV_DiscreteGridPointCoverage (from ISO 19123:2005)

Table 75 – Metadata of CV_DiscreteGridPointCoverage (class)

DEFINITION:	A coverage that returns the same feature attribute values for every direct position within any single spatial object, temporal object or spatiotemporal object in its domain.
SUBCLASS OF:	CV_DiscreteCoverage
STEREOTYPE:	«type»

Table 76 – Associations of CV_DiscreteGridPointCoverage (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
element	CV_GridPointValuePair [1..*]	
valueAssignment	CV_GridValuesMatrix [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.3. DirectPosition (from ISO 19107:2003)

Table 77 – Metadata of DirectPosition (class)

DEFINITION:	DirectPosition object data types (ISO 19107:2003, Figure 14) hold the coordinates for a position within some coordinate reference system. The coordinate reference system is described in ISO 19111. Since DirectPositions, as data types, will often be included in larger objects (such as GM_Objects) that have references to ISO19111:SC_CRS, the DirectPosition::coordinateReferenceSystem may be left NULL if this particular DirectPosition is included in a larger object with such a reference to a SC_CRS. In this case, the DirectPosition::coordinateReferenceSystem is implicitly assumed to take on the value of the containing object's SC_CRS.
SUBCLASS OF:	None
STEREOTYPE:	None

Table 78 – Associations of DirectPosition (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
CRS	CRS [0..1]	
CRS	SC_CRS [0..1]	

Table 79 – Attributes of DirectPosition (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
coordinate	Sequence <term number not resolved via ID number > [1..1]	
dimension	Integer [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.4. GM_Object (from ISO 19107:2003)

Table 80 – Metadata of GM_Object (class)

DEFINITION: GM_Object is the root class of the geometric object taxonomy and supports interfaces common to all geographically referenced geometric objects. GM_Object instances are sets of direct positions in a particular coordinate reference system. A GM_Object can be regarded as an infinite set of points that satisfies the set operation interfaces for a set of direct positions, TransfiniteSet<DirectPosition>. Since an infinite collection class cannot be implemented directly, a Boolean test for inclusion shall be provided by the GM_Object interface. This international standard concentrates on vector geometry classes, but future work may use GM_Object as a root class without modification.

NOTE: As a type, GM_Object does not have a well-defined default state or value representation as a data type. Instantiated subclasses of GM_Object will.

SUBCLASS OF:	none
STEREOTYPE:	«type»
CONSTRAINT:	dimension() > boundary().dimension (Invariant):
CONSTRAINT:	boundary().notEmpty() implies boundary().dimension() = dimension() -1 (Invariant):
CONSTRAINT:	boundary().isEmpty() = isCycle() (Invariant):

Table 81 – Associations of GM_Object (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
	Geometry [1..1]	
	TransfiniteSet <Direct Position> [1..1]	
	CV_DomainObject [1..1]	
CRS	CRS [0..1]	
CRS	SC_CRS [0..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.5. GM_MultiCurve (from ISO 19107:2003)

Table 82 – Metadata of GM_MultiCurve (class)

DEFINITION:	An aggregate class containing only instances of GM_OrientableCurve. The association role "element" shall be the set of GM_OrientableCurves contained in this GM_MultiCurve.
SUBCLASS OF:	GM_MultiPrimitive
STEREOTYPE:	«type»

Table 83 – Attributes of GM_MultiCurve (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
length	Length [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.6. GM_MultiPoint (from ISO 19107:2003)

Table 84 – Metadata of GM_MultiPoint (class)

DEFINITION:	GM_MultiPoint is an aggregate class containing only points. The association role "element" shall be the set of GM_Points contained in this GM_MultiPoint.
SUBCLASS OF:	GM_MultiPrimitive
STEREOTYPE:	«type»

Table 85 – Attributes of GM_MultiPoint (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
position	Set <DirectPosition> [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.7. GM_MultiSurface (from ISO 19107:2003)

Table 86 – Metadata of GM_MultiSurface (class)

DEFINITION:	An aggregate class containing only instances of GM_OrientableSurface. The association role "element" shall be the set of GM_OrientableSurfaces contained in this GM_MultiSurface.
-------------	---

SUBCLASS OF: GM_MultiPrimitive

STEREOTYPE: «type»

Table 87 – Attributes of GM_MultiSurface (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
area	Area [1..1]	
perimeter	Length [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.8. GM_Point (from ISO 19107:2003)

Table 88 – Metadata of GM_Point (class)

DEFINITION: GM_Point is the basic data type for a geometric object consisting of one and only one point.

SUBCLASS OF: GM_Primitive

STEREOTYPE: «type»

Table 89 – Associations of GM_Point (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
	Point [1..1]	
composite	GM_CompositePoint [0..*]	

Table 90 – Attributes of GM_Point (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
position	DirectPosition [1..1]	<p>The attribute “position” shall be the DirectPosition of this GM_Point.</p> <p>NOTE: In most cases, the state of a GM_Point is fully determined by its position attribute. The only exception to this is if the GM_Point has</p>

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
		been subclassed to provide additional non-geometric information such as symbology.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.9. GM_Solid (from ISO 19107:2003)

Table 91 – Metadata of GM_Solid (class)

DEFINITION:	GM_Solid, a subclass of GM_Primitive, is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.
SUBCLASS OF:	GM_Primitive
STEREOTYPE:	«type»

Table 92 – Associations of GM_Solid (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
composite	GM_CompositeSolid [0..*]	Solid [1..1]

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.10. GM_Surface (from ISO 19107:2003)

Table 93 – Metadata of GM_Surface (class)

DEFINITION:	GM_Surface is a subclass of GM_Primitive and is the basis for 2-dimensional geometry. Unorientable surfaces such as the Möbius band are not allowed. The orientation of a surface chooses an “up” direction through the choice of the upward normal, which, if the surface is not a cycle, is the side of the surface from which the exterior boundary appears counterclockwise. Reversal of the surface orientation reverses the curve orientation of each boundary component, and interchanges the conceptual “up” and “down” direction of the surface. If the surface is the boundary of a solid, the “up” direction is usually outward. For closed surfaces, which have no boundary, the up direction is that of the surface patches, which must be consistent with one another. Its included GM_SurfacePatches describe the interior structure of a GM_Surface.
-------------	--

NOTE: Other than the restriction on orientability, no other “validity” condition is required for GM_Surface.

SUBCLASS OF:	GM_OrientableSurface
---------------------	----------------------

| **STEREOTYPE:** | «type» |

Table 94 – Associations of GM_Surface (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
	GM_GenericSurface [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.11. GM_Tin (from ISO 19107:2003)

Table 95 – Metadata of GM_Tin (class)

DEFINITION:	A GM_Tin is a GM_TriangulatedSurface that uses the Delaunay algorithm or a similar algorithm complemented with consideration for breaklines, stoplines and maximum length of triangle sides (ISO 19107:2003, Figure 22). These networks satisfy the Delaunay criterion away from the modifications: For each triangle in the network, the circle passing through its vertexes does not contain, in its interior, the vertex of any other triangle.
--------------------	--

| **SUBCLASS OF:** | GM_TriangulatedSurface |
| **STEREOTYPE:** | «type» |

Table 96 – Attributes of GM_Tin (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
breakLines	Set <GM_LineString> [1..1]	
controlPoint	GM_Position [3..*]	
maxLength	Distance [1..1]	
stopLines	Set <GM_LineString> [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.12. GM_TriangulatedSurface (from ISO 19107:2003)

Table 97 – Metadata of GM_TriangulatedSurface (class)

DEFINITION: A GM_TriangulatedSurface is a GM_PolyhedralSurface that is composed only of triangles (GM_Triangle). There is no restriction on how the triangulation is derived.

SUBCLASS OF: GM_PolyhedralSurface

STEREOTYPE: «type»

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.13. SC_CRS (from ISO 19111:2019)

Table 98 – Metadata of SC_CRS (class)

DEFINITION: Coordinate reference system which is usually single but may be compound.

SUBCLASS OF: IO_IdentifiedObjectBase, RS_ReferenceSystem

STEREOTYPE: «type»

Table 99 – Associations of SC_CRS (class)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
coordOperation To	CC_Coordinate Operation [0..*]	Not-navigable association from a Coordinate Operation that uses this CRS as its targetCRS.
grid	CV_ReferenceableGrid [0..*]	

Table 100 – Attributes of SC_CRS (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
scope	CharacterString [1..*]	Description of usage, or limitations of usage, for which this CRS is valid. If unknown, enter "not known".

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.1.14. TM_Position (from ISO 19108:2006)

Table 101 – Metadata of TM_Position (class)

DEFINITION:	TM_Position is a union class that consists of one of the data types listed as its attributes. Date, Time, and DateTime are basic data types defined in ISO/TS 19103.
SUBCLASS OF:	None
STEREOTYPE:	«Union»

Table 102 – Attributes of TM_Position (class)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
anyOther	TM_TemporalPosition [1..1]	
date8601	Date [1..1]	
time8601	Time [1..1]	
dateTime8601	DateTime [1..1]	

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2. Core

Table 103 – Metadata of Core (ApplicationSchema)

DESCRIPTION:	The Core module defines the basic components of the CityGML conceptual model. This includes abstract base classes that define the core properties of more specialized thematic classes defined in other modules as well as concrete classes that are common to other modules, for example basic data types.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.2.1. Classes

8.2.1.1. AbstractAppearance

Table 104 – Metadata of AbstractAppearance (FeatureType)

DEFINITION: AbstractAppearance is the abstract superclass to represent any kind of appearance objects.

SUBCLASS OF: AbstractFeatureWithLifespan

STEREOTYPE: «FeatureType»

Table 105 – Attributes of AbstractAppearance (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractAppearance	ADEOfAbstractAppearance [0..*]	Augments AbstractAppearance with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.2. AbstractCityObject

Table 106 – Metadata of AbstractCityObject (FeatureType)

DEFINITION: AbstractCityObject is the abstract superclass of all thematic classes within the CityGML Conceptual Model.

SUBCLASS OF: AbstractFeatureWithLifespan

STEREOTYPE: «FeatureType»

Table 107 – Associations of AbstractCityObject (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
appearance	AbstractAppearance [1..*]	Relates appearances to the city object.
genericAttribute	AbstractGenericAttribute [1..*]	Relates generic attributes to the city object.

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
generalizesTo	AbstractCityObject [1..*]	Relates generalized representations of the same real-world object in different Levels of Detail to the city object. The direction of this relation is from the city object to the corresponding generalized city objects.
external Reference	ExternalReference [1..*]	References external objects in other information systems that have a relation to the city object.
relatedTo	AbstractCityObject [1..*]	Relates other city objects to the city object. It also describes how the city objects are related to each other.
dynamizer	AbstractDynamizer [1..*]	Relates Dynamizer objects to the city object. These allow timeseries data to override static attribute values of the city object.

Table 108 – Attributes of AbstractCityObject (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract CityObject	ADEOfAbstractCity Object [0..*]	Augments AbstractCityObject with properties defined in an ADE.
relativeToTerrain	RelativeToTerrain [0..1]	Describes the vertical position of the city object relative to the surrounding terrain.
relativeToWater	RelativeToWater [0..1]	Describes the vertical position of the city object relative to the surrounding water surface.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.3. AbstractDynamizer

Table 109 – Metadata of AbstractDynamizer (FeatureType)

DEFINITION:	AbstractDynamizer is the abstract superclass to represent Dynamizer objects.
SUBCLASS OF:	AbstractFeatureWithLifespan
STEREOTYPE:	«FeatureType»

Table 110 – Attributes of AbstractDynamizer (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Dynamizer	ADEOfAbstract Dynamizer [0..*]	Augments AbstractDynamizer with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.4. AbstractFeature

Table 111 – Metadata of AbstractFeature (FeatureType)

DEFINITION:	AbstractFeature is the abstract superclass of all feature types within the CityGML Conceptual Model.
SUBCLASS OF:	AnyFeature
STEREOTYPE:	«FeatureType»

Table 112 – Attributes of AbstractFeature (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractFeature	ADEOfAbstractFeature [0..*]	Augments AbstractFeature with properties defined in an ADE.
description	CharacterString [0..1]	Provides further information on the feature.
featureID	ID [1..1]	Specifies the unique identifier of the feature that is valid in the instance document within which it occurs.
identifier	ScopedName [0..1]	Specifies the unique identifier of the feature that is valid globally.
name	GenericName [0..*]	Specifies the name of the feature.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.5. AbstractFeatureWithLifespan

Table 113 – Metadata of AbstractFeatureWithLifespan (FeatureType)

DEFINITION:	AbstractFeatureWithLifespan is the base class for all CityGML features. This class allows the optional specification of the real-world and database times for the existence of each feature.
SUBCLASS OF:	AbstractFeature
STEREOTYPE:	«FeatureType»

Table 114 – Attributes of AbstractFeatureWithLifespan (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractFeatureWithLifespan	ADEOfAbstractFeatureWithLifespan [0..*]	Augments AbstractFeatureWithLifespan with properties defined in an ADE.
creationDate	DateTime [0..1]	Indicates the date at which a CityGML feature was added to the City Model.
terminationDate	DateTime [0..1]	Indicates the date at which a CityGML feature was removed from the CityModel.
validFrom	DateTime [0..1]	Indicates the date at which a CityGML feature started to exist in the real world.
validTo	DateTime [0..1]	Indicates the date at which a CityGML feature ended to exist in the real world.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.6. AbstractLogicalSpace

Table 115 – Metadata of AbstractLogicalSpace (FeatureType)

DEFINITION:	AbstractLogicalSpace is the abstract superclass for all types of logical spaces. Logical space refers to spaces that are not bounded by physical surfaces but are defined according to thematic considerations.
SUBCLASS OF:	AbstractSpace
STEREOTYPE:	«FeatureType»

Table 116 – Attributes of AbstractLogicalSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractLogicalSpace	ADEOfAbstractLogicalSpace [0..*]	Augments AbstractLogicalSpace with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.7. AbstractOccupiedSpace

Table 117 – Metadata of AbstractOccupiedSpace (FeatureType)

DEFINITION:	AbstractOccupiedSpace is the abstract superclass for all types of physically occupied spaces. Occupied space refers to spaces that are partially or entirely filled with matter.
SUBCLASS OF:	AbstractPhysicalSpace
STEREOTYPE:	«FeatureType»

Table 118 – Associations of AbstractOccupiedSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
lod3ImplicitRepresentation	ImplicitGeometry [0..1]	Relates to an implicit geometry that represents the occupied space in Level of Detail 3.
lod2ImplicitRepresentation	ImplicitGeometry [0..1]	Relates to an implicit geometry that represents the occupied space in Level of Detail 2.
lod1ImplicitRepresentation	ImplicitGeometry [0..1]	Relates to an implicit geometry that represents the occupied space in Level of Detail 1.

Table 119 – Attributes of AbstractOccupiedSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractOccupiedSpace	ADEOfAbstractOccupiedSpace [0..*]	Augments AbstractOccupiedSpace with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.8. AbstractPhysicalSpace

Table 120 – Metadata of AbstractPhysicalSpace (FeatureType)

DEFINITION:	AbstractPhysicalSpace is the abstract superclass for all types of physical spaces. Physical space refers to spaces that are fully or partially bounded by physical objects.
SUBCLASS OF:	AbstractSpace
STEREOTYPE:	«FeatureType»

Table 121 – Associations of AbstractPhysicalSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
lod3TerrainIntersectionCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 3.
pointCloud	AbstractPointCloud [0..1]	Relates to a 3D PointCloud that represents the physical space.
lod1TerrainIntersectionCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 1.
lod2TerrainIntersectionCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the terrain intersection curve of the physical space in Level of Detail 2.

Table 122 – Attributes of AbstractPhysicalSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractPhysicalSpace	ADEOfAbstractPhysicalSpace [0..*]	Augments AbstractPhysicalSpace with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.9. AbstractPointCloud

Table 123 – Metadata of AbstractPointCloud (FeatureType)

DEFINITION:	AbstractPointCloud is the abstract superclass to represent PointCloud objects.
SUBCLASS OF:	AbstractFeature
STEREOTYPE:	«FeatureType»

Table 124 – Attributes of AbstractPointCloud (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractPointCloud	ADEOfAbstractPointCloud [0..*]	Augments AbstractPointCloud with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.10. AbstractSpace

Table 125 – Metadata of AbstractSpace (FeatureType)

DEFINITION:	AbstractSpace is the abstract superclass for all types of spaces. A space is an entity of volumetric extent in the real world.
SUBCLASS OF:	AbstractCityObject
STEREOTYPE:	«FeatureType»

Table 126 – Associations of AbstractSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
lod2MultiCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the space in Level of Detail 2.
lod3MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the space in Level of Detail 3.
lod0MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the space in Level of Detail 0.
lod1Solid	GM_Solid [0..1]	Relates to a 3D Solid geometry that represents the space in Level of Detail 1.
lod3Solid	GM_Solid [0..1]	Relates to a 3D Solid geometry that represents the space in Level of Detail 3.
boundary	AbstractSpace Boundary [1..*]	Relates to surfaces that bound the space.
lod0MultiCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the space in Level of Detail 0.
lod2Solid	GM_Solid [0..1]	Relates to a 3D Solid geometry that represents the space in Level of Detail 2.
lod0Point	GM_Point [0..1]	Relates to a 3D Point geometry that represents the space in Level of Detail 0.
lod3MultiCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the space in Level of Detail 3.
lod2MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the space in Level of Detail 2.

Table 127 – Attributes of AbstractSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractSpace	ADEOfAbstractSpace [0..*]	Augments AbstractSpace with properties defined in an ADE.
area	QualifiedArea [0..*]	Specifies qualified areas related to the space.
spaceType	SpaceType [0..1]	Specifies the degree of openness of a space.
volume	QualifiedVolume [0..*]	Specifies qualified volumes related to the space.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.11. AbstractSpaceBoundary

Table 128 – Metadata of AbstractSpaceBoundary (FeatureType)

DEFINITION:	AbstractSpaceBoundary is the abstract superclass for all types of space boundaries. A space boundary is an entity with areal extent in the real world. Space boundaries are objects that bound a Space. They also realize the contact between adjacent spaces.
SUBCLASS OF:	AbstractCityObject
STEREOTYPE:	«FeatureType»

Table 129 – Attributes of AbstractSpaceBoundary (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractSpaceBoundary	ADEOfAbstractSpaceBoundary [0..*]	Augments AbstractSpaceBoundary with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.12. AbstractThematicSurface

Table 130 – Metadata of AbstractThematicSurface (FeatureType)

DEFINITION:	AbstractThematicSurface is the abstract superclass for all types of thematic surfaces.
SUBCLASS OF:	AbstractSpaceBoundary

STEREOTYPE: «FeatureType»

Table 131 – Associations of AbstractThematicSurface (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
lod3MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 3.
lod2MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 2.
pointCloud	AbstractPointCloud [0..1]	Relates to a 3D PointCloud that represents the thematic surface.
lod0MultiCurve	GM_MultiCurve [0..1]	Relates to a 3D MultiCurve geometry that represents the thematic surface in Level of Detail 0.
lod0MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 0.
lod1MultiSurface	GM_MultiSurface [0..1]	Relates to a 3D MultiSurface geometry that represents the thematic surface in Level of Detail 1.

Table 132 – Attributes of AbstractThematicSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractThematicSurface	ADEOfAbstractThematicSurface [0..*]	Augments AbstractThematicSurface with properties defined in an ADE.
area	QualifiedArea [0..*]	Specifies qualified areas related to the thematic surface.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.13. AbstractUnoccupiedSpace

Table 133 – Metadata of AbstractUnoccupiedSpace (FeatureType)

DEFINITION:	AbstractUnoccupiedSpace is the abstract superclass for all types of physically unoccupied spaces. Unoccupied space refers to spaces that are entirely or mostly free of matter.
SUBCLASS OF:	AbstractPhysicalSpace
STEREOTYPE:	«FeatureType»

Table 134 – Attributes of AbstractUnoccupiedSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Unoccupied Space	ADEOfAbstract UnoccupiedSpace [0..*]	Augments AbstractUnoccupiedSpace with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.14. AbstractVersion

Table 135 – Metadata of AbstractVersion (FeatureType)

DEFINITION: AbstractVersion is the abstract superclass to represent Version objects.

SUBCLASS OF: AbstractFeatureWithLifespan

STEREOTYPE: «FeatureType»

Table 136 – Attributes of AbstractVersion (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Version	ADEOfAbstractVersion [0..*]	Augments AbstractVersion with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.15. AbstractVersionTransition

Table 137 – Metadata of AbstractVersionTransition (FeatureType)

DEFINITION: AbstractVersionTransition is the abstract superclass to represent VersionTransition objects.

SUBCLASS OF: AbstractFeatureWithLifespan

STEREOTYPE: «FeatureType»

Table 138 – Attributes of AbstractVersionTransition (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractVersion	ADEOfAbstractVersion	
Transition [0..*]		Augments AbstractVersionTransition with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.16. Address

Table 139 – Metadata of Address (FeatureType)

DEFINITION: Address represents an address of a city object.

SUBCLASS OF: AbstractFeature

STEREOTYPE: «FeatureType»

Table 140 – Associations of Address (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
multiPoint	GM_MultiPoint [0..1]	Relates to the MultiPoint geometry of the Address. The geometry relates the address spatially to a city object.
xalAddress	XALAddress [1..1]	Relates an OASIS address object to the Address.

Table 141 – Attributes of Address (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAddress	ADEOfAddress [0..*]	Augments the Address with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.17. CityModel

Table 142 – Metadata of CityModel (FeatureType)

DEFINITION:	CityModel is the container for all objects belonging to a city model.
SUBCLASS OF:	AbstractFeatureWithLifespan
STEREOTYPE:	«FeatureType»

Table 143 – Associations of CityModel (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
cityModel Member	CityModelMember [1..*]	Relates to all objects that are part of the CityModel.

Table 144 – Attributes of CityModel (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfCityModel	ADEOfCityModel [0..*]	Augments the CityModel with properties defined in an ADE.
engineeringCRS	EngineeringCRS [0..1]	Specifies the local engineering coordinate reference system of the City Model that can be provided inline the CityModel instead of referencing a well-known CRS definition. 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.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.18. CityObjectRelation

Table 145 – Metadata of CityObjectRelation (ObjectType)

DEFINITION:	CityObjectRelation represents a specific relation from the city object in which the relation is included to another city object.
SUBCLASS OF:	None
STEREOTYPE:	«ObjectType»

Table 146 – Associations of CityObjectRelation (ObjectType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
genericAttribute	AbstractGenericAttribute [1..*]	Relates generic attributes to the CityObjectRelation.

Table 147 – Attributes of CityObjectRelation (ObjectType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
relationType	RelationTypeValue [1..1]	Indicates the specific type of the CityObjectRelation.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.19. ClosureSurface

Table 148 – Metadata of ClosureSurface (FeatureType)

DEFINITION:	ClosureSurface is a special type of thematic surface used to close holes in volumetric objects. Closure surfaces are virtual (non-physical) surfaces.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«FeatureType»

Table 149 – Attributes of ClosureSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfClosureSurface	ADEOfClosureSurface [0..*]	Augments the ClosureSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.1.20. ImplicitGeometry

Table 150 – Metadata of ImplicitGeometry (ObjectType)

DEFINITION:	ImplicitGeometry is a geometry representation where the shape is stored only once as a prototypical geometry. Examples are a tree or other vegetation object, a traffic light or a traffic sign. This prototypic geometry object can be re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model.
-------------	---

SUBCLASS OF:	None
---------------------	------

| **STEREOTYPE:** | «ObjectType» |

Table 151 – Associations of ImplicitGeometry (ObjectType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
relative Geometry	GM_Object [0..1]	Relates to a prototypical geometry in a local coordinate system stored inline with the city model.
referencePoint	GM_Point [1..1]	Relates to a 3D Point geometry that represents the base point of the object in the world coordinate system.
appearance	AbstractAppearance [1..*]	Relates appearances to the ImplicitGeometry.

Table 152 – Attributes of ImplicitGeometry (ObjectType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
libraryObject	URI [0..1]	Specifies the URI that points to the prototypical geometry stored in an external file.
MimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external file that stores the prototypical geometry.
objectID	ID [1..1]	Specifies the unique identifier of the ImplicitGeometry.
transformation Matrix	Transformation Matrix4x4 [1..1]	Specifies the mathematical transformation (translation, rotation, and scaling) between the prototypical geometry and the actual spatial position of the object.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2. Basic types

8.2.2.1. Code

Table 153 – Metadata of Code (BasicType)

DEFINITION:	Code is a basic type for a String-based term, keyword, or name that can additionally have a code space.
--------------------	---

| **SUBCLASS OF:** | None |

STEREOTYPE: «BasicType»

Table 154 – Attributes of Code (BasicType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
codeSpace	URI [0..1]	Associates the Code with an authority that controls the term, keyword, or name.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2.2. DoubleBetween0and1

Table 155 – Metadata of DoubleBetween0and1 (BasicType)

DEFINITION: DoubleBetween0and1 is a basic type for values, which are greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

CONSTRAINT: valueBetween0and1:
inv: DoubleBetween0and1.allInstances() -> forAll(p | p >= 0 and p <= 1)

8.2.2.3. DoubleBetween0and1List

Table 156 – Metadata of DoubleBetween0and1List (BasicType)

DEFINITION: DoubleBetween0and1List is a basic type that represents a list of double values greater or equal than 0 and less or equal than 1. The type is used for color encoding, for example.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

Table 157 – Attributes of DoubleBetween0and1List (BasicType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
list	DoubleBetween0and1 [1..1]	Specifies the list of double values.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2.4. DoubleList

Table 158 – Metadata of DoubleList (BasicType)

DEFINITION: DoubleList is an ordered sequence of double values.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

Table 159 – Attributes of DoubleList (BasicType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
list	Real [1..1]	Specifies the list of double values.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2.5. DoubleOrNilReasonList

Table 160 – Metadata of DoubleOrNilReasonList (BasicType)

DEFINITION: DoubleOrNilReasonList is a basic type that represents a list of double values and/or nil reasons.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

Table 161 – Attributes of DoubleOrNilReasonList (BasicType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
list	DoubleOrNilReason [1..1]	Specifies the list of double values and/or nil reasons.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2.6. ID

Table 162 – Metadata of ID (BasicType)

DEFINITION: ID is a basic type that represents a unique identifier.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

8.2.2.7. IntegerBetween0and3

Table 163 – Metadata of IntegerBetween0and3 (BasicType)

DEFINITION: IntegerBetween0and3 is a basic type for integer values, which are greater or equal than 0 and less or equal than 3. The type is used for encoding the LOD number.

SUBCLASS OF: None

STEREOTYPE: «BasicType»

CONSTRAINT: valueBetween0and3:
inv: IntegerBetween0and3.allInstances()->forAll(p | p >= 0 and p <= 3)

8.2.2.8. MeasureOrNilReasonList

Table 164 – Metadata of MeasureOrNilReasonList (BasicType)

DEFINITION: MeasureOrNilReasonList is a basic type that represents a list of double values and/or nil reasons together with a unit of measurement.

SUBCLASS OF: DoubleOrNilReasonList

STEREOTYPE: «BasicType»

Table 165 – Attributes of MeasureOrNilReasonList (BasicType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
uom	UnitOfMeasure [1..1]	Specifies the unit of measurement of the double values.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.2.9. TransformationMatrix2×2

Table 166 – Metadata of TransformationMatrix2×2 (BasicType)

DEFINITION:	TransformationMatrix2×2 is a 2 by 2 matrix represented as a list of four double values in row major order.
SUBCLASS OF:	DoubleList
STEREOTYPE:	«BasicType»
CONSTRAINT:	<code>lengthOfList: inv: list->size() = 4</code>

8.2.2.10. TransformationMatrix3×4

Table 167 – Metadata of TransformationMatrix3×4 (BasicType)

DEFINITION:	TransformationMatrix3×4 is a 3 by 4 matrix represented as a list of twelve double values in row major order.
SUBCLASS OF:	DoubleList
STEREOTYPE:	«BasicType»
CONSTRAINT:	<code>lengthOfList: inv: list->size() = 12</code>

8.2.2.11. TransformationMatrix4×4

Table 168 – Metadata of TransformationMatrix4×4 (BasicType)

DEFINITION:	TransformationMatrix4×4 is a 4 by 4 matrix represented as a list of sixteen double values in row major order.
-------------	---

SUBCLASS OF:	DoubleList
STEREOTYPE:	«BasicType»
CONSTRAINT:	<pre>lengthOfList: inv: list->size() = 16</pre>

8.2.3. Unions

8.2.3.1. CityModelMember

Table 169 – Metadata of CityModelMember (Union)

DEFINITION:	CityModelMember is a union type that enumerates the different types of objects that can occur as members of a city model.
SUBCLASS OF:	None
STEREOTYPE:	«Union»

Table 170 – Attributes of CityModelMember (Union)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
appearanceMember	AbstractAppearance [1..1]	Specifies the appearances of the CityModel.
cityObjectMember	AbstractCityObject [1..1]	Specifies the city objects that are part of the CityModel.
featureMember	AbstractFeature [1..1]	Specifies the feature objects that are part of the CityModel. It allows to include objects that are not derived from a class defined in the CityGML conceptual model, but from the ISO 19109 class AnyFeature.
versionMember	AbstractVersion [1..1]	Specifies the different versions of the CityModel.
versionTransitionMember	AbstractVersionTransition [1..1]	Specifies the transitions between the different versions of the City Model.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.3.2. DoubleOrNilReason

Table 171 – Metadata of DoubleOrNilReason (Union)

DEFINITION:	DoubleOrNilReason is a union type that allows for choosing between a double value and a nil reason.
SUBCLASS OF:	None
STEREOTYPE:	«Union»

Table 172 – Attributes of DoubleOrNilReason (Union)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
nilReason	NilReason [1..1]	Specifies the nil reason.
value	Real [1..1]	Specifies the double value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.3.3. NilReason

Table 173 – Metadata of NilReason (Union)

DEFINITION:	NilReason is a union type that allows for choosing between two different types of nil reason.
SUBCLASS OF:	None
STEREOTYPE:	«Union»

Table 174 – Attributes of NilReason (Union)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
nilReason Enumeration	NilReasonEnumeration [1..1]	Indicates a nil reason that is provided in a code list.
URI	URI [1..1]	Specifies a URI that points to a resource that describes the nil reason.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.4. Code lists

8.2.4.1. IntervalValue

Table 175 – Metadata of IntervalValue (CodeList)

DEFINITION: IntervalValue is a code list used to specify a time period.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.2.4.2. MimeTypeValue

Table 176 – Metadata of MimeTypeValue (CodeList)

DEFINITION: MimeTypeValue is a code list used to specify the MIME type of a referenced resource.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.2.4.3. NilReasonEnumeration

Table 177 – Metadata of NilReasonEnumeration (CodeList)

DEFINITION: NilReasonEnumeration is a code list that enumerates the different nil reasons.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.2.4.4. OccupantTypeValue

Table 178 – Metadata of OccupantTypeValue (CodeList)

DEFINITION:	OccupantTypeValue is a code list used to classify occupants.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.2.4.5. OtherRelationTypeValue

Table 179 – Metadata of OtherRelationTypeValue (CodeList)

DEFINITION:	OtherRelationTypeValue is a code list used to classify other types of city object relations.
SUBCLASS OF:	RelationTypeValue
STEREOTYPE:	«CodeList»

8.2.4.6. QualifiedAreaTypeValue

Table 180 – Metadata of QualifiedAreaTypeValue (CodeList)

DEFINITION:	QualifiedAreaTypeValue is a code list used to specify area types.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.2.4.7. QualifiedVolumeTypeValue

Table 181 – Metadata of QualifiedVolumeTypeValue (CodeList)

DEFINITION:	QualifiedVolumeTypeValue is a code list used to specify volume types.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.2.4.8. RelationTypeValue

Table 182 – Metadata of RelationTypeValue (CodeList)

DEFINITION: RelationTypeValue is a code list used to classify city object relations.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.2.4.9. TemporalRelationTypeValue

Table 183 – Metadata of TemporalRelationTypeValue (CodeList)

DEFINITION: TemporalRelationTypeValue is a code list used to classify temporal city object relations.

SUBCLASS OF: RelationTypeValue

STEREOTYPE: «CodeList»

8.2.4.10. TopologicalRelationTypeValue

Table 184 – Metadata of TopologicalRelationTypeValue (CodeList)

DEFINITION: TopologicalRelationTypeValue is a code list used to classify topological city object relations.

SUBCLASS OF: RelationTypeValue

STEREOTYPE: «CodeList»

8.2.5. Data types

8.2.5.1. AbstractGenericAttribute

Table 185 – Metadata of AbstractGenericAttribute (DataType)

DEFINITION:	AbstractGenericAttribute is the abstract superclass for all types of generic attributes.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.2. ADEOfAbstractAppearance

Table 186 – Metadata of ADEOfAbstractAppearance (DataType)

DEFINITION:	ADEOfAbstractAppearance acts as a hook to define properties within an ADE that are to be added to AbstractAppearance.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.3. ADEOfAbstractCityObject

Table 187 – Metadata of ADEOfAbstractCityObject (DataType)

DEFINITION:	ADEOfAbstractCityObject acts as a hook to define properties within an ADE that are to be added to AbstractCityObject.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.4. ADEOfAbstractDynamizer

Table 188 – Metadata of ADEOfAbstractDynamizer (DataType)

DEFINITION:	ADEOfAbstractDynamizer acts as a hook to define properties within an ADE that are to be added to AbstractDynamizer.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.5. ADEOfAbstractFeature

Table 189 – Metadata of ADEOfAbstractFeature (DataType)

DEFINITION:	ADEOfAbstractFeature acts as a hook to define properties within an ADE that are to be added to AbstractFeature.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.6. ADEOfAbstractFeatureWithLifespan

Table 190 – Metadata of ADEOfAbstractFeatureWithLifespan (DataType)

DEFINITION:	ADEOfAbstractFeatureWithLifespan acts as a hook to define properties within an ADE that are to be added to AbstractFeatureWithLifespan.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.7. ADEOfAbstractLogicalSpace

Table 191 – Metadata of ADEOfAbstractLogicalSpace (DataType)

DEFINITION:	ADEOfAbstractLogicalSpace acts as a hook to define properties within an ADE that are to be added to AbstractLogicalSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.8. ADEOfAbstractOccupiedSpace

Table 192 – Metadata of ADEOfAbstractOccupiedSpace (DataType)

DEFINITION:	ADEOfAbstractOccupiedSpace acts as a hook to define properties within an ADE that are to be added to AbstractOccupiedSpace.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.5.9. ADEOfAbstractPhysicalSpace

Table 193 – Metadata of ADEOfAbstractPhysicalSpace (DataType)

DEFINITION: ADEOfAbstractPhysicalSpace acts as a hook to define properties within an ADE that are to be added to AbstractPhysicalSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.5.10. ADEOfAbstractPointCloud

Table 194 – Metadata of ADEOfAbstractPointCloud (DataType)

DEFINITION: ADEOfAbstractPointCloud acts as a hook to define properties within an ADE that are to be added to AbstractPointCloud.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.5.11. ADEOfAbstractSpace

Table 195 – Metadata of ADEOfAbstractSpace (DataType)

DEFINITION: ADEOfAbstractSpace acts as a hook to define properties within an ADE that are to be added to AbstractSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.5.12. ADEOfAbstractSpaceBoundary

Table 196 – Metadata of ADEOfAbstractSpaceBoundary (DataType)

DEFINITION:	ADEOfAbstractSpaceBoundary acts as a hook to define properties within an ADE that are to be added to AbstractSpaceBoundary.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.13. ADEOfAbstractThematicSurface

Table 197 – Metadata of ADEOfAbstractThematicSurface (DataType)

DEFINITION:	ADEOfAbstractThematicSurface acts as a hook to define properties within an ADE that are to be added to AbstractThematicSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.14. ADEOfAbstractUnoccupiedSpace

Table 198 – Metadata of ADEOfAbstractUnoccupiedSpace (DataType)

DEFINITION:	ADEOfAbstractUnoccupiedSpace acts as a hook to define properties within an ADE that are to be added to AbstractUnoccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.15. ADEOfAbstractVersion

Table 199 – Metadata of ADEOfAbstractVersion (DataType)

DEFINITION:	ADEOfAbstractVersion acts as a hook to define properties within an ADE that are to be added to AbstractVersion.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.16. ADEOfAbstractVersionTransition

Table 200 – Metadata of ADEOfAbstractVersionTransition (DataType)

DEFINITION:	ADEOfAbstractVersionTransition acts as a hook to define properties within an ADE that are to be added to AbstractVersionTransition.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.17. ADEOfAddress

Table 201 – Metadata of ADEOfAddress (DataType)

DEFINITION:	ADEOfAddress acts as a hook to define properties within an ADE that are to be added to an Address.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.18. ADEOfCityModel

Table 202 – Metadata of ADEOfCityModel (DataType)

DEFINITION:	ADEOfCityModel acts as a hook to define properties within an ADE that are to be added to a CityModel.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.2.5.19. ADEOfClosureSurface

Table 203 – Metadata of ADEOfClosureSurface (DataType)

DEFINITION:	ADEOfClosureSurface acts as a hook to define properties within an ADE that are to be added to a ClosureSurface.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.5.20. ExternalReference

Table 204 – Metadata of ExternalReference (DataType)

DEFINITION: ExternalReference is a reference to a corresponding object in another information system, for example in the German cadastre (ALKIS), the German topographic information system (ATKIS), or the OS UK MasterMap®.

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 205 – Attributes of ExternalReference (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
informationSystem	URI [0..1]	Specifies the URI that points to the external information system.
relationType	URI [0..1]	Specifies a URI that additionally qualifies the ExternalReference. The URI can point to a definition from an external ontology (e.g. the same As relation from OWL) and allows for mapping the ExternalReference to RDF triples.
targetResource	URI [1..1]	Specifies the URI that points to the object in the external information system.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.5.21. Occupancy

Table 206 – Metadata of Occupancy (DataType)

DEFINITION: Occupancy is an application-dependent indication of what is contained by a feature.

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 207 – Attributes of Occupancy (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
interval	IntervalValue [0..1]	Indicates the time period the occupants are contained by a feature.
number OfOccupants	Integer [1..1]	Indicates the number of occupants contained by a feature.
occupantType	OccupantTypeValue [0..1]	Indicates the specific type of the occupants that are contained by a feature.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.5.22. QualifiedArea

Table 208 – Metadata of QualifiedArea (DataType)

DEFINITION:	QualifiedArea is an application-dependent measure of the area of a space or of a thematic surface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 209 – Attributes of QualifiedArea (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
area	Area [1..1]	Specifies the value of the QualifiedArea.
typeOfArea	QualifiedAreaType Value [1..1]	Indicates the specific type of the QualifiedArea.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.5.23. QualifiedVolume

Table 210 – Metadata of QualifiedVolume (DataType)

DEFINITION:	QualifiedVolume is an application-dependent measure of the volume of a space.
SUBCLASS OF:	None

STEREOTYPE: «DataType»

Table 211 – Attributes of QualifiedVolume (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
typeOfVolume	QualifiedVolumeType Value [1..1]	Indicates the specific type of the QualifiedVolume.
volume	Volume [1..1]	Specifies the value of the QualifiedVolume.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.2.5.24. XALAddress

Table 212 – Metadata of XALAddress (DataType)

DEFINITION: XALAddress represents address details according to the OASIS xAL standard.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.2.6. Enumerations

8.2.6.1. RelativeToTerrain

Table 213 – Metadata of RelativeToTerrain (Enumeration)

DEFINITION: RelativeToTerrain enumerates the spatial relations of a city object relative to terrain in a qualitative way.

STEREOTYPE: «Enumeration»

Table 214 – Values of RelativeToTerrain (Enumeration)

LITERAL VALUE	DEFINITION
entirelyAboveTerrain	Indicates that the city object is located entirely above the terrain.

LITERAL VALUE	DEFINITION
substantiallyAboveTerrain	Indicates that the city object is for the most part located above the terrain.
substantiallyAboveAndBelowTerrain	Indicates that the city object is located half above the terrain and half below the terrain.
substantiallyBelowTerrain	Indicates that the city object is for the most part located below the terrain.
entirelyBelowTerrain	Indicates that the city object is located entirely below the terrain.

8.2.6.2. RelativeToWater

Table 215 – Metadata of RelativeToWater (Enumeration)

DEFINITION:	RelativeToWater enumerates the spatial relations of a city object relative to the water surface in a qualitative way.
STEREOTYPE:	«Enumeration»

Table 216 – Values of RelativeToWater (Enumeration)

LITERAL VALUE	DEFINITION
entirelyAboveWaterSurface	Indicates that the city object is located entirely above the water surface.
substantiallyAboveWaterSurface	Indicates that the city object is for the most part located above the water surface.
substantiallyAboveAndBelowWaterSurface	Indicates that the city object is located half above the water surface and half below the water surface.
substantiallyBelowWaterSurface	Indicates that the city object is for the most part located below the water surface.
entirelyBelowWaterSurface	Indicates that the city object is located entirely below the water surface.
temporarilyAboveAndBelowWaterSurface	Indicates that the city object is temporarily located above or below the water level, because the height of the water surface is varying.

8.2.6.3. SpaceType

Table 217 – Metadata of SpaceType (Enumeration)

DEFINITION: SpaceType is an enumeration that characterises a space according to its closure properties.

STEREOTYPE: «Enumeration»

Table 218 – Values of SpaceType (Enumeration)

LITERAL VALUE	DEFINITION
closed	Indicates that the space has boundaries at the bottom, at the top, and on all sides.
open	Indicates that the space has at maximum a boundary at the bottom.
semiOpen	Indicates that the space has a boundary at the bottom and on at least one side.

8.3. Appearance

Table 219 – Metadata of Appearance (ApplicationSchema)

DESCRIPTION: The Appearance module supports the modelling of the observable surface properties of City GML features in the form of textures and material.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.3.1. Classes

8.3.1.1. AbstractSurfaceData

Table 220 – Metadata of AbstractSurfaceData (FeatureType)

DEFINITION: AbstractSurfaceData is the abstract superclass for different kinds of textures and material.

SUBCLASS OF: AbstractFeature

STEREOTYPE: «FeatureType»

Table 221 – Attributes of AbstractSurfaceData (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractSurfaceData	ADEOfAbstractSurfaceData [0..*]	Augments AbstractSurfaceData with properties defined in an ADE.
isFront	Boolean [0..1]	Indicates whether the texture or material is assigned to the front side or the back side of the surface geometry object.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.2. AbstractTexture

Table 222 – Metadata of AbstractTexture (FeatureType)

DEFINITION:	AbstractTexture is the abstract superclass to represent the common attributes of the classes ParameterizedTexture and GeoreferencedTexture.
SUBCLASS OF:	AbstractSurfaceData
STEREOTYPE:	«FeatureType»

Table 223 – Attributes of AbstractTexture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractTexture	ADEOfAbstractTexture [0..*]	Augments AbstractTexture with properties defined in an ADE.
borderColor	ColorPlusOpacity [0..1]	Specifies the color of that part of the surface that is not covered by the texture.
imageURI	URI [1..1]	Specifies the URI that points to the external image data file.
contentType	MimeTypeValue [0..1]	Specifies the MIME type of the external point cloud file.
textureType	TextureType [0..1]	Indicates the specific type of the texture.
wrapMode	WrapMode [0..1]	Specifies the behaviour of the texture when the texture is smaller than the surface to which it is applied.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.3. Appearance

Table 224 – Metadata of Appearance (FeatureType)

DEFINITION:	An Appearance is a collection of surface data, i.e. observable properties for surface geometry objects in the form of textures and material.
SUBCLASS OF:	AbstractAppearance
STEREOTYPE:	«FeatureType»

Table 225 – Associations of Appearance (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
surfaceData	AbstractSurfaceData [1..*]	Relates to the surface data that are part of the Appearance.

Table 226 – Attributes of Appearance (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAppearance	ADEOfAppearance [0..*]	Augments the Appearance with properties defined in an ADE.
theme	CharacterString [0..1]	Specifies the topic of the Appearance. Each Appearance contains surface data for one theme only. Examples of themes are infrared radiation, noise pollution, or earthquake-induced structural stress.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.4. GeoreferencedTexture

Table 227 – Metadata of GeoreferencedTexture (FeatureType)

DEFINITION:	A GeoreferencedTexture is a texture that uses a planimetric projection. It contains an implicit parameterization that is either stored within the image file, an accompanying world file or specified using the orientation and referencePoint elements.
SUBCLASS OF:	AbstractTexture
STEREOTYPE:	«FeatureType»

Table 228 – Associations of GeoreferencedTexture (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
referencePoint	GM_Point [0..1]	Relates to the 2D Point geometry that represents the center of the upper left image pixel in world space.

Table 229 – Attributes of GeoreferencedTexture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGeoreferencedTexture	ADEOfGeoreferencedTexture [0..*]	Augments the GeoreferencedTexture with properties defined in an ADE.
orientation	TransformationMatrix2x2 [0..1]	Specifies the rotation and scaling of the image in form of a 2×2 matrix.
preferWorldFile	Boolean [0..1]	Indicates whether the georeference from the image file or the accompanying world file should be preferred.
target	URI [0..*]	Specifies the URI that points to the surface geometry objects to which the texture is applied.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.5. ParameterizedTexture

Table 230 – Metadata of ParameterizedTexture (FeatureType)

DEFINITION:	A ParameterizedTexture is a texture that uses texture coordinates or a transformation matrix for parameterization.
SUBCLASS OF:	AbstractTexture
STEREOTYPE:	«FeatureType»

Table 231 – Associations of ParameterizedTexture (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
textureParameterization	AbstractTextureParameterization [1..*]	Relates to the texture coordinates or transformation matrices used for parameterization.

Table 232 – Attributes of ParameterizedTexture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfParameterizedTexture	ADEOfParameterizedTexture [0..*]	Augments the ParameterizedTexture with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.6. TextureAssociation

Table 233 – Metadata of TextureAssociation (ObjectType)

DEFINITION: TextureAssociation denotes the relation of a texture to a surface geometry object.

SUBCLASS OF: None

STEREOTYPE: «ObjectType»

Table 234 – Attributes of TextureAssociation (ObjectType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
target	URI [1..1]	Specifies the URI that points to the surface geometry object to which the texture is applied.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.1.7. X3DMaterial

Table 235 – Metadata of X3DMaterial (FeatureType)

DEFINITION: X3DMaterial defines properties for surface geometry objects based on the material definitions from the X3D and COLLADA standards.

SUBCLASS OF: AbstractSurfaceData

STEREOTYPE: «FeatureType»

Table 236 – Attributes of X3DMaterial (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfX3DMaterial	ADEOfX3DMaterial [0..*]	Augments the X3DMaterial with properties defined in an ADE.
ambientIntensity	DoubleBetween0and1 [0..1]	Specifies the minimum percentage of diffuseColor that is visible regardless of light sources.
diffuseColor	Color [0..1]	Specifies the color of the light diffusely reflected by the surface geometry object.
emissiveColor	Color [0..1]	Specifies the color of the light emitted by the surface geometry object.
isSmooth	Boolean [0..1]	Specifies which interpolation method is used for the shading of the surface geometry object. If the attribute is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading).
shininess	DoubleBetween0and1 [0..1]	Specifies the sharpness of the specular highlight.
specularColor	Color [0..1]	Specifies the color of the light directly reflected by the surface geometry object.
target	URI [0..*]	Specifies the URI that points to the surface geometry objects to which the material is applied.
transparency	DoubleBetween0and1 [0..1]	Specifies the degree of transparency of the surface geometry object.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.2. Basic types

8.3.2.1. Color

Table 237 – Metadata of Color (BasicType)

DEFINITION: Color is a list of three double values between 0 and 1 defining an RGB color value.

SUBCLASS OF: DoubleBetween0and1List

STEREOTYPE: «BasicType»

CONSTRAINT: lengthOfList:
inv: list->size() = 3

8.3.2.2. ColorPlusOpacity

Table 238 – Metadata of ColorPlusOpacity (BasicType)

DEFINITION:	Color is a list of four double values between 0 and 1 defining an RGBA color value. Opacity value of 0 means transparent.
SUBCLASS OF:	DoubleBetween0and1List
STEREOTYPE:	«BasicType»
CONSTRAINT:	<code>lengthOfList: inv: list->size() = 3 or list->size() = 4</code>

8.3.3. Unions

None.

8.3.4. Code lists

None.

8.3.5. Data types

8.3.5.1. AbstractTextureParameterization

Table 239 – Metadata of AbstractTextureParameterization (DataType)

DEFINITION:	AbstractTextureParameterization is the abstract superclass for different kinds of texture parameterizations.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.2. ADEOfAbstractSurfaceData

Table 240 – Metadata of ADEOfAbstractSurfaceData (DataType)

DEFINITION:	ADEOfAbstractSurfaceData acts as a hook to define properties within an ADE that are to be added to AbstractSurfaceData.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.3. ADEOfAbstractTexture

Table 241 – Metadata of ADEOfAbstractTexture (DataType)

DEFINITION:	ADEOfAbstractTexture acts as a hook to define properties within an ADE that are to be added to AbstractTexture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.4. ADEOfAppearance

Table 242 – Metadata of ADEOfAppearance (DataType)

DEFINITION:	ADEOfAppearance acts as a hook to define properties within an ADE that are to be added to an Appearance.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.5. ADEOfGeoreferencedTexture

Table 243 – Metadata of ADEOfGeoreferencedTexture (DataType)

DEFINITION:	ADEOfGeoreferencedTexture acts as a hook to define properties within an ADE that are to be added to a GeoreferencedTexture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.6. ADEOfParameterizedTexture

Table 244 – Metadata of ADEOfParameterizedTexture (DataType)

DEFINITION:	ADEOfParameterizedTexture acts as a hook to define properties within an ADE that are to be added to a ParameterizedTexture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.7. ADEOfX3DMaterial

Table 245 – Metadata of ADEOfX3DMaterial (DataType)

DEFINITION:	ADEOfX3DMaterial acts as a hook to define properties within an ADE that are to be added to an X3DMaterial.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.3.5.8. TexCoordGen

Table 246 – Metadata of TexCoordGen (DataType)

DEFINITION:	TexCoordGen defines texture parameterization using a transformation matrix.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 247 – Associations of TexCoordGen (DataType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
crs	SC_CRS [0..1]	Relates to the coordinate reference system of the transformation matrix.

Table 248 – Attributes of TexCoordGen (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
worldToTexture	Transformation Matrix3×4 [1..1]	Specifies the 3×4 transformation matrix that defines the transformation between world coordinates and texture coordinates.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.5.9. TexCoordList

Table 249 – Metadata of TexCoordList (DataType)

DEFINITION: TexCoordList defines texture parameterization using texture coordinates.

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 250 – Attributes of TexCoordList (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
ring	URI [1..*]	Specifies the URIs that point to the LinearRings that are parameterized using the given texture coordinates.
texture Coordinates	DoubleList [1..*]	Specifies the coordinates of texture used for parameterization. The texture coordinates are provided separately for each LinearRing of the surface geometry object.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.3.6. Enumerations

8.3.6.1. TextureType

Table 251 – Metadata of TextureType (Enumeration)

DEFINITION: TextureType enumerates the different texture types.

STEREOTYPE: «Enumeration»

Table 252 – Values of TextureType (Enumeration)

LITERAL VALUE	DEFINITION
specific	Indicates that the texture is specific to a single surface.
typical	Indicates that the texture is characteristic of a surface and can be used repeatedly.
unknown	Indicates that the texture type is not known.

8.3.6.2. WrapMode

Table 253 – Metadata of WrapMode (Enumeration)

DEFINITION: WrapMode enumerates the different fill modes for textures.

STEREOTYPE: «Enumeration»

Table 254 – Values of WrapMode (Enumeration)

LITERAL VALUE	DEFINITION
none	Indicates that the texture is applied to the surface “as is”. The part of the surface that is not covered by the texture is shown fully transparent. [cf. COLLADA]
wrap	Indicates that the texture is repeated until the surface is fully covered. [cf. COLLADA]
mirror	Indicates that the texture is repeated and mirrored. [cf. COLLADA]
clamp	Indicates that the texture is stretched to the edges of the surface. [cf. COLLADA]
border	Indicates that the texture is applied to the surface “as is”. The part of the surface that is not covered by the texture is filled with the RGBA color that is specified in the attribute border Color. [cf. COLLADA]

8.4. CityFurniture

Table 255 – Metadata of CityFurniture (ApplicationSchema)

DESCRIPTION:	The CityFurniture module supports representation of city furniture objects. 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.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.4.1. Classes

8.4.1.1. CityFurniture

Table 256 – Metadata of CityFurniture (TopLevelFeatureType)

DEFINITION:	CityFurniture is an object or piece of equipment installed in the outdoor environment for various purposes. Examples include street signs, traffic signals, street lamps, benches, fountains.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 257 – Attributes of CityFurniture (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfCityFurniture	ADEOfCityFurniture [0..*]	Augments the CityFurniture with properties defined in an ADE.
class	CityFurnitureClass Value [0..1]	Indicates the specific type of the CityFurniture.
function	CityFurnitureFunction Value [0..*]	Specifies the intended purposes of the CityFurniture.
usage	CityFurnitureUsage Value [0..*]	Specifies the actual uses of the CityFurniture.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.4.2. Basic types

None.

8.4.3. Unions

None.

8.4.4. Code lists

8.4.4.1. CityFurnitureClassValue

Table 258 – Metadata of CityFurnitureClassValue (CodeList)

DEFINITION: CityFurnitureClassValue is a code list used to further classify a CityFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.4.4.2. CityFurnitureFunctionValue

Table 259 – Metadata of CityFurnitureFunctionValue (CodeList)

DEFINITION: CityFurnitureFunctionValue is a code list that enumerates the different purposes of a City Furniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.4.4.3. CityFurnitureUsageValue

Table 260 – Metadata of CityFurnitureUsageValue (CodeList)

DEFINITION: CityFurnitureUsageValue is a code list that enumerates the different uses of a CityFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.4.5. Data types

8.4.5.1. ADEOfCityFurniture

Table 261 – Metadata of ADEOfCityFurniture (DataType)

DEFINITION:	ADEOfCityFurniture acts as a hook to define properties within an ADE that are to be added to a CityFurniture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.4.6. Enumerations

None.

8.5. CityObjectGroup

Table 262 – Metadata of CityObjectGroup (ApplicationSchema)

DESCRIPTION:	The CityObjectGroup module supports grouping of city objects. Arbitrary city objects may be aggregated in groups according to user-defined criteria. A group may be further classified by application-specific attributes.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.5.1. Classes

8.5.1.1. CityObjectGroup

Table 263 – Metadata of CityObjectGroup (TopLevelFeatureType)

DEFINITION:	A CityObjectGroup represents an application-specific aggregation of city objects 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 member of a group may be qualified by a role name, reflecting the role each city object plays in the context of the group.
SUBCLASS OF:	AbstractLogicalSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 264 – Associations of CityObjectGroup (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
parent	AbstractCityObject [0..1]	Relates to a city object to which the CityObjectGroup belongs.
groupMember	AbstractCityObject [1..*]	Relates to the city objects that are part of the CityObjectGroup.

Table 265 – Attributes of CityObjectGroup (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfCityObjectGroup	ADEOfCityObjectGroup [0..*]	Augments the CityObjectGroup with properties defined in an ADE.
class	CityObjectGroupClass Value [0..1]	Indicates the specific type of the CityObjectGroup.
function	CityObjectGroupFunctionValue [0..*]	Specifies the intended purposes of the CityObjectGroup.
usage	CityObjectGroupUsageValue [0..*]	Specifies the actual usages of the CityObjectGroup.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.5.1.2. Role

Table 266 – Metadata of Role (ObjectType)

DEFINITION:	Role qualifies the function of a city object within the CityObjectGroup.
SUBCLASS OF:	None

STEREOTYPE: «ObjectType»

Table 267 – Attributes of Role (ObjectType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
role	CharacterString [0..1]	Describes the role the city object plays within the CityObjectGroup.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.5.2. Basic types

None.

8.5.3. Unions

None.

8.5.4. Code lists

8.5.4.1. CityObjectGroupClassValue

Table 268 – Metadata of CityObjectGroupClassValue (CodeList)

DEFINITION: CityObjectGroupClassValue is a code list used to further classify a CityObjectGroup.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.5.4.2. CityObjectGroupFunctionValue

Table 269 – Metadata of CityObjectGroupFunctionValue (CodeList)

DEFINITION: CityObjectGroupFunctionValue is a code list that enumerates the different purposes of a CityObjectGroup.

SUBCLASS OF: None

STEREOTYPE:	«CodeList»
-------------	------------

8.5.4.3. CityObjectGroupUsageValue

Table 270 – Metadata of CityObjectGroupUsageValue (CodeList)

DEFINITION:	CityObjectGroupUsageValue is a code list that enumerates the different uses of a CityObject Group.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.5.5. Data types

8.5.5.1. ADEOfCityObjectGroup

Table 271 – Metadata of ADEOfCityObjectGroup (DataType)

DEFINITION:	ADEOfCityObjectGroup acts as a hook to define properties within an ADE that are to be added to a CityObjectGroup.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.5.6. Enumerations

None.

8.6. Dynamizer

Table 272 – Metadata of Dynamizer (ApplicationSchema)

DESCRIPTION:	The Dynamizer module supports the injection of timeseries data for individual attributes of CityGML features. Timeseries data can either be retrieved from external Sensor APIs (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms),
--------------	--

external standardized timeseries files (e.g. OGC TimeseriesML or OGC Observations & Measurements), external tabulated files (e.g CSV) or can be represented inline as basic time-value pairs.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.6.1. Classes

8.6.1.1. AbstractAtomicTimeseries

Table 273 – Metadata of AbstractAtomicTimeseries (FeatureType)

DEFINITION: AbstractAtomicTimeseries represents the attributes and relationships that are common to all kinds of atomic timeseries (GenericTimeseries, TabulatedFileTimeseries, StandardFileTimeseries). An atomic timeseries represents time-varying data of a specific data type for a single contiguous time interval.

SUBCLASS OF: AbstractTimeseries

STEREOTYPE: «FeatureType»

Table 274 – Attributes of AbstractAtomicTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractAtomicTimeseries	ADEOfAbstractAtomicTimeseries [0..*]	Augments AbstractAtomicTimeseries with properties defined in an ADE.
observationProperty	CharacterString [1..1]	Specifies the phenomenon for which the atomic timeseries provides observation values.
uom	CharacterString [0..1]	Specifies the unit of measurement of the observation values.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.2. AbstractTimeseries

Table 275 – Metadata of AbstractTimeseries (FeatureType)

DEFINITION:	AbstractTimeseries is the abstract superclass representing any type of timeseries data.
SUBCLASS OF:	AbstractFeature
STEREOTYPE:	«FeatureType»

Table 276 – Attributes of AbstractTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractTimeseries	ADEOfAbstractTimeseries [0..*]	Augments AbstractTimeseries with properties defined in an ADE.
firstTimestamp	TM_Position [0..1]	Specifies the beginning of the timeseries.
lastTimestamp	TM_Position [0..1]	Specifies the end of the timeseries.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.3. CompositeTimeseries

Table 277 – Metadata of CompositeTimeseries (FeatureType)

DEFINITION:	A CompositeTimeseries is a (possibly recursive) aggregation of atomic and composite timeseries. The components of a composite timeseries must have non-overlapping time intervals.
SUBCLASS OF:	AbstractTimeseries
STEREOTYPE:	«FeatureType»

Table 278 – Associations of CompositeTimeseries (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
component	TimeseriesComponent [1..*]	Relates to the atomic and composite timeseries that are part of the CompositeTimeseries. The referenced timeseries are sequentially ordered.

Table 279 – Attributes of CompositeTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfCompositeTimeseries	ADEOfCompositeTimeseries [0..*]	Augments the CompositeTimeseries with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.4. Dynamizer

Table 280 – Metadata of Dynamizer (FeatureType)

DEFINITION:	A Dynamizer is an object that injects timeseries data for an individual attribute of the city object in which it is included. The timeseries data overrides the static value of the referenced city object attribute in order to represent dynamic (time-dependent) variations of its value.
SUBCLASS OF:	AbstractDynamizer
STEREOTYPE:	«FeatureType»

Table 281 – Associations of Dynamizer (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
dynamicData	AbstractTimeseries [0..1]	Relates to the timeseries data that is given either inline within a CityGML dataset or by a link to an external file containing timeseries data.
sensorConnection	SensorConnection [0..1]	Relates to the sensor API that delivers timeseries data.

Table 282 – Attributes of Dynamizer (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfDynamizer	ADEOfDynamizer [0..*]	Augments the Dynamizer with properties defined in an ADE.
attributeRef	CharacterString [1..1]	Specifies the attribute of a CityGML feature whose value is overridden or replaced by the (dynamic) values specified by the Dynamizer.
endTime	TM_Position [0..1]	Specifies the end of the time span for which the Dynamizer provides dynamic values.
startTime	TM_Position [0..1]	Specifies the beginning of the time span for which the Dynamizer provides dynamic values.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.5. GenericTimeseries

Table 283 – Metadata of GenericTimeseries (FeatureType)

DEFINITION:	A GenericTimeseries represents time-varying data in the form of embedded time-value-pairs of a specific data type for a single contiguous time interval.
SUBCLASS OF:	AbstractAtomicTimeseries
STEREOTYPE:	«FeatureType»
CONSTRAINT:	<pre> dataTypeOfValue: inv: if valueType = TimeseriesTypeValue::integer then TimeValuePair->forAll(c c.intValue->size()=1) else if valueType = TimeseriesTypeValue::double then TimeValuePair->forAll(c c.doubleValue->size()=1) else if valueType = TimeseriesTypeValue::string then TimeValuePair->forAll(c c.stringValue->size()=1) else if valueType = TimeseriesTypeValue::geometry then TimeValuePair->forAll(c c.geometryValue->size()=1) else if valueType = TimeseriesTypeValue::uri then TimeValuePair->forAll(c c.uriValue->size()=1) else if valueType = TimeseriesTypeValue::bool then TimeValuePair->forAll(c c.boolValue->size()=1) else if valueType = TimeseriesTypeValue::implicitGeometry then TimeValuePair->forAll(c c.implicitGeometryValue->size()=1) else TimeValuePair->forAll(c c.appearanceValue->size()=1) endif endif endif endif endif endif </pre>

Table 284 – Associations of GenericTimeseries (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
timeValuePair	TimeValuePair [1..*]	Relates to the time-value-pairs that are part of the GenericTimeseries.

Table 285 – Attributes of GenericTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGenericTimeseries	ADEOfGenericTimeseries [0..*]	Augments the GenericTimeseries with properties defined in an ADE.
valueType	TimeseriesTypeValue [1..1]	Indicates the specific type of all time-value-pairs that are part of the GenericTimeseries.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.6. StandardFileTimeseries

Table 286 – Metadata of StandardFileTimeseries (FeatureType)

DEFINITION:	A StandardFileTimeseries represents time-varying data for a single contiguous time interval. The data is provided in an external file referenced in the StandardFileTimeseries. The data within the external file is encoded according to a dedicated format for the representation of timeseries data such as using the OGC TimeseriesML or OGC Observations & Measurements Standard. The data type of the data has to be specified within the external file.
SUBCLASS OF:	AbstractAtomicTimeseries
STEREOTYPE:	«FeatureType»

Table 287 – Attributes of StandardFileTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfStandardFileTimeseries	ADEOfStandardFileTimeseries [0..*]	Augments the StandardFileTimeseries with properties defined in an ADE.
fileLocation	URI [1..1]	Specifies the URI that points to the external timeseries file.
fileType	StandardFileTypeValue [1..1]	Specifies the format used to represent the timeseries data.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external timeseries file.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.1.7. TabulatedFileTimeseries

Table 288 – Metadata of TabulatedFileTimeseries (FeatureType)

DEFINITION:	A TabulatedFileTimeseries represents time-varying data of a specific data type for a single contiguous time interval. The data is provided in an external file referenced in the TabulatedFileTimeseries. The file contains table structured data using an appropriate file format such as comma-separated values (CSV), Microsoft Excel (XLSX) or Google Spreadsheet. The timestamps and the values are given in specific columns of the table. Each row represents a single time-value-pair. A subset of rows can be selected using the idColumn and idValue attributes.
-------------	--

SUBCLASS OF: AbstractAtomicTimeseries

STEREOTYPE: «FeatureType»

CONSTRAINT:

```
columnNumberOrColumnName:  
inv: (timeColumnNo->notEmpty() or timeColumnName->notEmpty())  
and (valueColumnNo->notEmpty() or valueColumnName-  
>notEmpty()) and (idValue->notEmpty() implies idColumnName-  
>notEmpty() or idColumnName->notEmpty())
```

Table 289 – Attributes of TabulatedFileTimeseries (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTabulatedFileTimeseries	ADEOfTabulatedFileTimeseries [0..*]	Augments the TabulatedFileTimeseries with properties defined in an ADE.
decimalSymbol	Character [0..1]	Indicates which symbol is used to separate the integer part from the fractional part of a decimal number.
fieldSeparator	CharacterString [1..1]	Indicates which symbol is used to separate the individual values in the tabulated file.
fileLocation	URI [1..1]	Specifies the URI that points to the external timeseries file.
fileType	TabulatedFileType Value [1..1]	Specifies the format used to represent the timeseries data.
idColumnName	CharacterString [0..1]	Specifies the name of the column that stores the identifier of the time-value-pair.
idColumnNo	Integer [0..1]	Specifies the number of the column that stores the identifier of the time-value-pair.
idValue	CharacterString [0..1]	Specifies the value of the identifier for which the time-value-pairs are to be selected.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external timeseries file.
numberOfHeaderLines	Integer [0..1]	Indicates the number of lines at the beginning of the tabulated file that represent headers.
timeColumnName	CharacterString [0..1]	Specifies the name of the column that stores the timestamp of the time-value-pair.
timeColumnNo	Integer [0..1]	Specifies the number of the column that stores the timestamp of the time-value-pair.
valueColumnName	CharacterString [0..1]	Specifies the name of the column that stores the value of the time-value-pair.
valueColumnNo	Integer [0..1]	Specifies the number of the column that stores the value of the time-value-pair.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
valueType	TimeseriesTypeValue [1..1]	Indicates the specific type of the timeseries data.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.2. Basic types

None.

8.6.3. Unions

None.

8.6.4. Code lists

8.6.4.1. AuthenticationTypeValue

Table 290 – Metadata of AuthenticationTypeValue (CodeList)

DEFINITION:	AuthenticationTypeValue is a code list used to specify the authentication method to be used to access the referenced sensor service. Each value provides enough information such that a software application could determine the required access credentials.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.6.4.2. SensorConnectionTypeValue

Table 291 – Metadata of SensorConnectionTypeValue (CodeList)

DEFINITION:	SensorConnectionTypeValue is a code list used to specify the type of the referenced sensor service. Each value provides enough information such that a software application would be able to identify the API type and version.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.6.4.3. StandardFileTypeValue

Table 292 – Metadata of StandardFileTypeValue (CodeList)

DEFINITION:	StandardFileTypeValue is a code list used to specify the type of the referenced external timeseries data file. Each value provides information about the standard and version.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.6.4.4. TabulatedFileTypeValue

Table 293 – Metadata of TabulatedFileTypeValue (CodeList)

DEFINITION:	TabulatedFileTypeValue is a code list used to specify the data format of the referenced external tabulated data file.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.6.5. Data types

8.6.5.1. ADEOfAbstractAtomicTimeseries

Table 294 – Metadata of ADEOfAbstractAtomicTimeseries (DataType)

DEFINITION:	ADEOfAbstractAtomicTimeseries acts as a hook to define properties within an ADE that are to be added to AbstractAtomicTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.2. ADEOfAbstractTimeseries

Table 295 – Metadata of ADEOfAbstractTimeseries (DataType)

DEFINITION:	ADEOfAbstractTimeseries acts as a hook to define properties within an ADE that are to be added to AbstractTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.3. ADEOfCompositeTimeseries

Table 296 – Metadata of ADEOfCompositeTimeseries (DataType)

DEFINITION:	ADEOfCompositeTimeseries acts as a hook to define properties within an ADE that are to be added to a CompositeTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.4. ADEOfDynamizer

Table 297 – Metadata of ADEOfDynamizer (DataType)

DEFINITION:	ADEOfDynamizer acts as a hook to define properties within an ADE that are to be added to a Dynamizer.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.5. ADEOfGenericTimeseries

Table 298 – Metadata of ADEOfGenericTimeseries (DataType)

DEFINITION:	ADEOfGenericTimeseries acts as a hook to define properties within an ADE that are to be added to a GenericTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.6. ADEOfStandardFileTimeseries

Table 299 – Metadata of ADEOfStandardFileTimeseries (DataType)

DEFINITION:	ADEOfStandardFileTimeseries acts as a hook to define properties within an ADE that are to be added to a StandardFileTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.7. ADEOfTabulatedFileTimeseries

Table 300 – Metadata of ADEOfTabulatedFileTimeseries (DataType)

DEFINITION:	ADEOfTabulatedFileTimeseries acts as a hook to define properties within an ADE that are to be added to a TabulatedFileTimeseries.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.6.5.8. SensorConnection

Table 301 – Metadata of SensorConnection (DataType)

DEFINITION:	A SensorConnection provides all details that are required to retrieve a specific datastream from an external sensor web service. This data type comprises the service type (e.g. OGC SensorThings API, OGC Sensor Observation Services, MQTT, proprietary platforms), the URL of the sensor service, the identifier for the sensor or thing, and its observed property as well as information about the required authentication method.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 302 – Associations of SensorConnection (DataType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
sensorLocation	AbstractCityObject [0..1]	Relates the sensor to the city object where it is located.

Table 303 – Attributes of SensorConnection (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
authType	AuthenticationType Value [0..1]	Specifies the type of authentication required to be able to access the Sensor API.
baseURL	URI [0..1]	Specifies the base URL of the Sensor API request.
connectionType	SensorConnectionType Value [1..1]	Indicates the type of Sensor API to which the SensorConnection refers.
datastreamID	CharacterString [0..1]	Specifies the datastream that is retrieved by the SensorConnection.
linkToObservation	CharacterString [0..1]	Specifies the complete URL to the observation request.
linkToSensorDescription	CharacterString [0..1]	Specifies the complete URL to the sensor description request.
mqttServer	CharacterString [0..1]	Specifies the name of the MQTT Server. This attribute is relevant when the MQTT Protocol is used to connect to a Sensor API.
mqttTopic	CharacterString [0..1]	Names the specific datastream that is retrieved by the Sensor Connection. This attribute is relevant when the MQTT Protocol is used to connect to a Sensor API.
observationID	CharacterString [0..1]	Specifies the unique identifier of the observation that is retrieved by the SensorConnection.
observationProperty	CharacterString [1..1]	Specifies the phenomenon for which the SensorConnection provides observations.
sensorID	CharacterString [0..1]	Specifies the unique identifier of the sensor from which the Sensor Connection retrieves observations.
sensorName	CharacterString [0..1]	Specifies the name of the sensor from which the SensorConnection retrieves observations.
uom	CharacterString [0..1]	Specifies the unit of measurement of the observations.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.5.9. TimeseriesComponent

Table 304 – Metadata of TimeseriesComponent (DataType)

DEFINITION: TimeseriesComponent represents an element of a CompositeTimeseries.

SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 305 – Associations of TimeseriesComponent (DataType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
timeseries	AbstractTimeseries [1..1]	Relates a timeseries to the TimeseriesComponent.

Table 306 – Attributes of TimeseriesComponent (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
additionalGap	TM_Duration [0..1]	Specifies how much extra time is added after all repetitions as an additional gap.
repetitions	Integer [1..1]	Specifies how often the timeseries that is referenced by the Timeseries Component should be iterated.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.5.10. TimeValuePair

Table 307 – Metadata of TimeValuePair (DataType)

DEFINITION: A TimeValuePair represents a value that is valid for a given timepoint. For each TimeValuePair, only one of the value properties can be used mutually exclusive. Which value property has to be provided depends on the selected value type in the GenericTimeSeries feature, in which the TimeValuePair is included.

SUBCLASS OF:	None
STEREOTYPE:	«DataType»

CONSTRAINT:

```

singleValue:
inv: intValue->size() + doubleValue->size() + stringValue-
>size() + geometryValue->size() + uriValue->size() +
boolValue->size() + implicitGeometryValue->size() +
appearanceValue->size() = 1

```

Table 308 – Attributes of TimeValuePair (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
appearanceValue	AbstractAppearance [0..1]	Specifies the “Appearance” value of the TimeValuePair.
boolValue	Boolean [0..1]	Specifies the “Boolean” value of the TimeValuePair.
doubleValue	Real [0..1]	Specifies the “Double” value of the TimeValuePair.
geometryValue	GM_Object [0..1]	Specifies the geometry value of the TimeValuePair.
implicit GeometryValue	ImplicitGeometry [0..1]	Specifies the “ImplicitGeometry” value of the TimeValuePair.
intValue	Integer [0..1]	Specifies the “Integer” value of the TimeValuePair.
stringValue	CharacterString [0..1]	Specifies the “String” value of the TimeValuePair.
timestamp	TM_Position [1..1]	Specifies the timepoint at which the value of the TimeValuePair is valid.
uriValue	URI [0..1]	Specifies the “URI” value of the TimeValuePair.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.6.6. Enumerations

8.6.6.1. TimeseriesTypeValue

Table 309 – Metadata of TimeseriesTypeValue (Enumeration)

DEFINITION: TimeseriesTypeValue enumerates the possible value types for GenericTimeseries and TimeValuePair.

STEREOTYPE: «Enumeration»

Table 310 – Values of TimeseriesTypeValue (Enumeration)

LITERAL VALUE	DEFINITION
int	Indicates that the values of the GenericTimeseries are of type “Integer”.
double	Indicates that the values of the GenericTimeseries are of type “Double”.
string	Indicates that the values of the GenericTimeseries are of type “String”.
geometry	Indicates that the values of the GenericTimeseries are geometries.
uri	Indicates that the values of the GenericTimeseries are of type “URI”.
bool	Indicates that the values of the GenericTimeseries are of type “Boolean”.
implicitGeometry	Indicates that the values of the GenericTimeseries are of type “ImplicitGeometry”.
appearance	Indicates that the values of the GenericTimeseries are of type “Appearance”.

8.7. Generics

Table 311 – Metadata of Generics (ApplicationSchema)

DESCRIPTION:	The Generics module supports application-specific extensions to the CityGML conceptual model. These extensions may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. Generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.7.1. Classes

8.7.1.1. GenericLogicalSpace

Table 312 – Metadata of GenericLogicalSpace (TopLevelFeatureType)

DEFINITION:	A GenericLogicalSpace is a space that is not represented by any explicitly modelled AbstractLogicalSpace subclass within CityGML.
SUBCLASS OF:	AbstractLogicalSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 313 – Attributes of GenericLogicalSpace (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGenericLogicalSpace	ADEOfGenericLogicalSpace [0..*]	Augments the GenericLogicalSpace with properties defined in an ADE.
class	GenericLogicalSpace ClassValue [0..1]	Indicates the specific type of the GenericLogicalSpace.
function	GenericLogicalSpace FunctionValue [0..*]	Specifies the intended purposes of the GenericLogicalSpace.
usage	GenericLogicalSpace UsageValue [0..*]	Specifies the actual uses of the GenericLogicalSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.1.2. GenericOccupiedSpace

Table 314 – Metadata of GenericOccupiedSpace (TopLevelFeatureType)

DEFINITION:	A GenericOccupiedSpace is a space that is not represented by any explicitly modelled AbstractOccupiedSpace subclass within CityGML.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 315 – Attributes of GenericOccupiedSpace (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGenericOccupiedSpace	ADEOfGenericOccupiedSpace [0..*]	Augments the GenericOccupiedSpace with properties defined in an ADE.
class	GenericOccupiedSpace ClassValue [0..1]	Indicates the specific type of the GenericOccupiedSpace.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
function	GenericOccupiedSpace FunctionValue [0..*]	Specifies the intended purposes of the GenericOccupiedSpace.
usage	GenericOccupiedSpace UsageValue [0..*]	Specifies the actual uses of the GenericOccupiedSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.1.3. GenericThematicSurface

Table 316 – Metadata of GenericThematicSurface (TopLevelFeatureType)

DEFINITION:	A GenericThematicSurface is a surface that is not represented by any explicitly modelled AbstractThematicSurface subclass within CityGML.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«TopLevelFeatureType»

Table 317 – Attributes of GenericThematicSurface (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGeneric ThematicSurface	ADEOfGeneric ThematicSurface [0..*]	Augments the GenericThematicSurface with properties defined in an ADE.
class	GenericThematic SurfaceClassValue [0..1]	Indicates the specific type of the GenericThematicSurface.
function	GenericThematic SurfaceFunctionValue [0..*]	Specifies the intended purposes of the GenericThematicSurface.
usage	GenericThematic SurfaceUsageValue [0..*]	Specifies the actual uses of the GenericThematicSurface.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.1.4. GenericUnoccupiedSpace

Table 318 – Metadata of GenericUnoccupiedSpace (TopLevelFeatureType)

DEFINITION:	A GenericUnoccupiedSpace is a space that is not represented by any explicitly modelled AbstractUnoccupiedSpace subclass within CityGML.
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 319 – Attributes of GenericUnoccupiedSpace (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGenericUnoccupiedSpace	ADEOfGenericUnoccupiedSpace [0..*]	Augments the GenericUnoccupiedSpace with properties defined in an ADE.
class	GenericUnoccupiedSpaceClassValue [0..1]	Indicates the specific type of the GenericUnoccupiedSpace.
function	GenericUnoccupiedSpaceFunctionValue [0..*]	Specifies the intended purposes of the GenericUnoccupiedSpace.
usage	GenericUnoccupiedSpaceUsageValue [0..*]	Specifies the actual uses of the GenericUnoccupiedSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.2. Basic types

None.

8.7.3. Unions

None.

8.7.4. Code lists

8.7.4.1. GenericLogicalSpaceClassValue

Table 320 – Metadata of GenericLogicalSpaceClassValue (CodeList)

DEFINITION:	GenericLogicalSpaceClassValue is a code list used to further classify a GenericLogicalSpace.
--------------------	--

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.7.4.2. GenericLogicalSpaceFunctionValue

Table 321 – Metadata of GenericLogicalSpaceFunctionValue (CodeList)

DEFINITION: GenericLogicalSpaceFunctionValue is a code list that enumerates the different purposes of a GenericLogicalSpace.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.7.4.3. GenericLogicalSpaceUsageValue

Table 322 – Metadata of GenericLogicalSpaceUsageValue (CodeList)

DEFINITION: GenericLogicalSpaceUsageValue is a code list that enumerates the different uses of a GenericLogicalSpace.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.7.4.4. GenericOccupiedSpaceClassValue

Table 323 – Metadata of GenericOccupiedSpaceClassValue (CodeList)

DEFINITION: GenericOccupiedSpaceClassValue is a code list used to further classify a GenericOccupied Space.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.7.4.5. GenericOccupiedSpaceFunctionValue

Table 324 – Metadata of GenericOccupiedSpaceFunctionValue (CodeList)

DEFINITION:	GenericOccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericOccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.6. GenericOccupiedSpaceUsageValue

Table 325 – Metadata of GenericOccupiedSpaceUsageValue (CodeList)

DEFINITION:	GenericOccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericOccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.7. GenericThematicSurfaceClassValue

Table 326 – Metadata of GenericThematicSurfaceClassValue (CodeList)

DEFINITION:	GenericThematicSurfaceClassValue is a code list used to further classify a GenericThematic Surface.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.8. GenericThematicSurfaceFunctionValue

Table 327 – Metadata of GenericThematicSurfaceFunctionValue (CodeList)

DEFINITION:	GenericThematicSurfaceFunctionValue is a code list that enumerates the different purposes of a GenericThematicSurface.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.9. GenericThematicSurfaceUsageValue

Table 328 – Metadata of GenericThematicSurfaceUsageValue (CodeList)

DEFINITION:	GenericThematicSurfaceUsageValue is a code list that enumerates the different uses of a GenericThematicSurface.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.10. GenericUnoccupiedSpaceClassValue

Table 329 – Metadata of GenericUnoccupiedSpaceClassValue (CodeList)

DEFINITION:	GenericUnoccupiedSpaceClassValue is a code list used to further classify a GenericUnoccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.11. GenericUnoccupiedSpaceFunctionValue

Table 330 – Metadata of GenericUnoccupiedSpaceFunctionValue (CodeList)

DEFINITION:	GenericUnoccupiedSpaceFunctionValue is a code list that enumerates the different purposes of a GenericUnoccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.7.4.12. GenericUnoccupiedSpaceUsageValue

Table 331 – Metadata of GenericUnoccupiedSpaceUsageValue (CodeList)

DEFINITION:	GenericUnoccupiedSpaceUsageValue is a code list that enumerates the different uses of a GenericUnoccupiedSpace.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.7.5. Data types

8.7.5.1. ADEOfGenericLogicalSpace

Table 332 – Metadata of ADEOfGenericLogicalSpace (DataType)

DEFINITION: ADEOfGenericLogicalSpace acts as a hook to define properties within an ADE that are to be added to a GenericLogicalSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.7.5.2. ADEOfGenericOccupiedSpace

Table 333 – Metadata of ADEOfGenericOccupiedSpace (DataType)

DEFINITION: ADEOfGenericOccupiedSpace acts as a hook to define properties within an ADE that are to be added to a GenericOccupiedSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.7.5.3. ADEOfGenericThematicSurface

Table 334 – Metadata of ADEOfGenericThematicSurface (DataType)

DEFINITION: ADEOfGenericThematicSurface acts as a hook to define properties within an ADE that are to be added to a GenericThematicSurface.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.7.5.4. ADEOfGenericUnoccupiedSpace

Table 335 – Metadata of ADEOfGenericUnoccupiedSpace (DataType)

DEFINITION:	ADEOfGenericUnoccupiedSpace acts as a hook to define properties within an ADE that are to be added to a GenericUnoccupiedSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.7.5.5. CodeAttribute

Table 336 – Metadata of CodeAttribute (DataType)

DEFINITION:	CodeAttribute is a data type used to define generic attributes of type “Code”.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 337 – Attributes of CodeAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the CodeAttribute.
value	Code [1..1]	Specifies the “Code” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.6. DateAttribute

Table 338 – Metadata of DateAttribute (DataType)

DEFINITION:	DateAttribute is a data type used to define generic attributes of type “Date”.
SUBCLASS OF:	None

STEREOTYPE: «DataType»

Table 339 – Attributes of DateAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the DateAttribute.
value	Date [1..1]	Specifies the “Date” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.7. DoubleAttribute

Table 340 – Metadata of DoubleAttribute (DataType)

DEFINITION:	DoubleAttribute is a data type used to define generic attributes of type “Double”.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 341 – Attributes of DoubleAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the DoubleAttribute.
value	Real [1..1]	Specifies the “Double” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.8. GenericAttributeSet

Table 342 – Metadata of GenericAttributeSet (DataType)

DEFINITION:	A GenericAttributeSet is a named collection of generic attributes.
SUBCLASS OF:	None

STEREOTYPE: «DataType»

Table 343 – Associations of GenericAttributeSet (DataType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
genericAttribute	AbstractGenericAttribute [1..*]	Relates to the generic attributes that are part of the GenericAttributeSet.

Table 344 – Attributes of GenericAttributeSet (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
codeSpace	URI [0..1]	Associates the GenericAttributeSet with an authority that maintains the collection of generic attributes.
name	CharacterString [1..1]	Specifies the name of the GenericAttributeSet.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.9. IntAttribute

Table 345 – Metadata of IntAttribute (DataType)

DEFINITION: IntAttribute is a data type used to define generic attributes of type “Integer”.

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 346 – Attributes of IntAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the IntAttribute.
value	Integer [1..1]	Specifies the “Integer” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.10. MeasureAttribute

Table 347 – Metadata of MeasureAttribute (DataType)

DEFINITION:	MeasureAttribute is a data type used to define generic attributes of type “Measure”.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 348 – Attributes of MeasureAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the MeasureAttribute.
value	Measure [1..1]	Specifies the value of the MeasureAttribute. The value is of type “Measure”, which can additionally provide the units of measure. [cf. ISO 19103]

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.11. StringAttribute

Table 349 – Metadata of StringAttribute (DataType)

DEFINITION:	StringAttribute is a data type used to define generic attributes of type “String”.	
SUBCLASS OF:	None	
STEREOTYPE:	«DataType»	

Table 350 – Attributes of StringAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the StringAttribute.
value	CharacterString [1..1]	Specifies the “String” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.5.12. UriAttribute

Table 351 – Metadata of UriAttribute (DataType)

DEFINITION: UriAttribute is a data type used to define generic attributes of type “URI”.

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 352 – Attributes of UriAttribute (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
name	CharacterString [1..1]	Specifies the name of the UriAttribute.
value	URI [1..1]	Specifies the “URI” value.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.7.6. Enumerations

None.

8.8. LandUse

Table 353 – Metadata of LandUse (ApplicationSchema)

DESCRIPTION: The LandUse module supports representation of areas of the earth's surface dedicated to a specific land use.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.8.1. Classes

8.8.1.1. LandUse

Table 354 – Metadata of LandUse (TopLevelFeatureType)

DEFINITION:	A LandUse object is an area of the earth's surface dedicated to a specific land use or having a specific land cover with or without vegetation, such as sand, rock, mud flats, forest, grasslands, or wetlands.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«TopLevelFeatureType»

Table 355 – Attributes of LandUse (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfLandUse	ADEOfLandUse [0..*]	Augments the LandUse with properties defined in an ADE.
class	LandUseClassValue [0..1]	Indicates the specific type of the LandUse.
function	LandUseFunctionValue [0..*]	Specifies the intended purposes of the LandUse.
usage	LandUseUsageValue [0..*]	Specifies the actual uses of the LandUse.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.8.2. Basic types

None.

8.8.3. Unions

None.

8.8.4. Code lists

8.8.4.1. LandUseClassValue

Table 356 – Metadata of LandUseClassValue (CodeList)

DEFINITION: LandUseClassValue is a code list used to further classify a LandUse.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.8.4.2. LandUseFunctionValue

Table 357 – Metadata of LandUseFunctionValue (CodeList)

DEFINITION: LandUseFunctionValue is a code list that enumerates the different purposes of a LandUse.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.8.4.3. LandUseUsageValue

Table 358 – Metadata of LandUseUsageValue (CodeList)

DEFINITION: LandUseUsageValue is a code list that enumerates the different uses of a LandUse.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.8.5. Data types

8.8.5.1. ADEOfLandUse

Table 359 – Metadata of ADEOfLandUse (DataType)

DEFINITION:	ADEOfLandUse acts as a hook to define properties within an ADE that are to be added to a LandUse.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.8.6. Enumerations

None.

8.9. PointCloud

Table 360 – Metadata of PointCloud (ApplicationSchema)

DESCRIPTION:	The PointCloud module supports representation of CityGML features by a collection of points.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.9.1. Classes

8.9.1.1. PointCloud

Table 361 – Metadata of PointCloud (FeatureType)

DEFINITION:	A PointCloud is an unordered collection of points that is a sampling of the geometry of a space or space boundary.
-------------	--

SUBCLASS OF: AbstractPointCloud

STEREOTYPE: «FeatureType»

CONSTRAINT:

```
inlineOrExternalPointCloud:  
inv: (points->notEmpty() and mimeType->isEmpty() and  
pointFile->isEmpty() and pointFileSrsName->isEmpty()) xor  
(points->isEmpty() and mimeType->notEmpty() and pointFile-  
>notEmpty())
```

Table 362 – Associations of PointCloud (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
points	GM_MultiPoint [0..1]	Relates to the 3D MultiPoint geometry of the PointCloud stored inline with the city model.

Table 363 – Attributes of PointCloud (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfPointCloud	ADEOfPointCloud [0..*]	Augments the PointCloud with properties defined in an ADE.
mimeType	MimeTypeValue [0..1]	Specifies the MIME type of the external point cloud file.
pointFile	URI [0..1]	Specifies the URI that points to the external point cloud file.
pointFileSrsName	CharacterString [0..1]	Indicates the coordinate reference system used by the external point cloud file.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.9.2. Basic types

None.

8.9.3. Unions

None.

8.9.4. Code lists

None.

8.9.5. Data types

8.9.5.1. ADEOfPointCloud

Table 364 – Metadata of ADEOfPointCloud (DataType)

DEFINITION:	ADEOfPointCloud acts as a hook to define properties within an ADE that are to be added to a PointCloud.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.9.6. Enumerations

None.

8.10. Relief

Table 365 – Metadata of Relief (ApplicationSchema)

DESCRIPTION:	The Relief module supports representation of the terrain. CityGML supports terrain representations at different levels of detail, reflecting different accuracies or resolutions. Terrain may be specified as a regular raster or grid, as a TIN, by break lines, and/or by mass points.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.10.1. Classes

8.10.1.1. AbstractReliefComponent

Table 366 – Metadata of AbstractReliefComponent (FeatureType)

DEFINITION:	An AbstractReliefComponent represents an element of the terrain surface – either a TIN, a raster or grid, mass points or break lines.
SUBCLASS OF:	AbstractSpaceBoundary
STEREOTYPE:	«FeatureType»
CONSTRAINT:	<pre>polygonGeometry: inv: extent.patch->size()=1 and extent.patch->forAll(oclIsKindOf(GM_Polygon))</pre>

Table 367 – Associations of AbstractReliefComponent (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
extent	GM_Surface [0..1]	Indicates the geometrical extent of the terrain component. The geometrical extent is provided as a 2D Surface geometry.

Table 368 – Attributes of AbstractReliefComponent (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractReliefComponent	ADEOfAbstractReliefComponent [0..*]	Augments AbstractReliefComponent with properties defined in an ADE.
lod	IntegerBetween0and3 [1..1]	Indicates the Level of Detail of the terrain component.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.1.2. BreaklineRelief

Table 369 – Metadata of BreaklineRelief (FeatureType)

DEFINITION:	A BreaklineRelief represents a terrain component with 3D lines. These lines denote break lines or ridge/valley lines.
SUBCLASS OF:	AbstractReliefComponent
STEREOTYPE:	«FeatureType»

Table 370 – Associations of BreaklineRelief (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
ridgeOrValley Lines	GM_MultiCurve [0..1]	Relates to the 3D MultiCurve geometry of the MassPointRelief. This association role is used to represent ridge or valley lines.
breaklines	GM_MultiCurve [0..1]	Relates to the 3D MultiCurve geometry of the MassPointRelief. This association role is used to represent break lines.

Table 371 – Attributes of BreaklineRelief (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBreakline Relief	ADEOfBreaklineRelief [0..*]	Augments the BreaklineRelief with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.1.3. MassPointRelief

Table 372 – Metadata of MassPointRelief (FeatureType)

DEFINITION:	A MassPointRelief represents a terrain component as a collection of 3D points.
SUBCLASS OF:	AbstractReliefComponent
STEREOTYPE:	«FeatureType»

Table 373 – Associations of MassPointRelief (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
pointCloud	AbstractPointCloud [0..1]	Relates to the 3D PointCloud of the MassPointRelief.
reliefPoints	GM_MultiPoint [0..1]	Relates to the 3D MultiPoint geometry of the MassPointRelief.

Table 374 – Attributes of MassPointRelief (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfMassPoint Relief	ADEOfMassPointRelief [0..*]	Augments the MassPointRelief with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.1.4. RasterRelief

Table 375 – Metadata of RasterRelief (FeatureType)

DEFINITION: A RasterRelief represents a terrain component as a regular raster or grid.

SUBCLASS OF: AbstractReliefComponent

STEREOTYPE: «FeatureType»

Table 376 – Associations of RasterRelief (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
grid	CV_DiscreteGridPoint Coverage [1..1]	Relates to the DiscreteGridPointCoverage of the RasterRelief.

Table 377 – Attributes of RasterRelief (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfRaster Relief	ADEOfRasterRelief [0..*]	Augments the RasterRelief with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.1.5. ReliefFeature

Table 378 – Metadata of ReliefFeature (TopLevelFeatureType)

DEFINITION: A ReliefFeature is a collection of terrain components representing the Earth's surface, also known as the Digital Terrain Model.

SUBCLASS OF: AbstractSpaceBoundary

STEREOTYPE: «TopLevelFeatureType»

Table 379 – Associations of ReliefFeature (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
reliefComponent	AbstractReliefComponent [1..*]	Relates to the terrain components that are part of the ReliefFeature.

Table 380 – Attributes of ReliefFeature (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfReliefFeature	ADEOfReliefFeature [0..*]	Augments the ReliefFeature with properties defined in an ADE.
lod	IntegerBetween0and3 [1..1]	Indicates the Level of Detail of the ReliefFeature.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.1.6. TINRelief

Table 381 – Metadata of TINRelief (FeatureType)

DEFINITION:	A TINRelief represents a terrain component as a triangulated irregular network.
SUBCLASS OF:	AbstractReliefComponent
STEREOTYPE:	«FeatureType»

Table 382 – Associations of TINRelief (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
tin	GM_TriangulatedSurface [1..1]	Relates to the triangulated surface of the TINRelief.

Table 383 – Attributes of TINRelief (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTINRelief	ADEOfTINRelief [0..*]	Augments the TINRelief with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.10.2. Basic types

None.

8.10.3. Unions

None.

8.10.4. Code lists

None.

8.10.5. Data types

8.10.5.1. ADEOfAbstractReliefComponent

Table 384 – Metadata of ADEOfAbstractReliefComponent (DataType)

DEFINITION:	ADEOfAbstractReliefComponent acts as a hook to define properties within an ADE that are to be added to AbstractReliefComponent.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.10.5.2. ADEOfBreaklineRelief

Table 385 – Metadata of ADEOfBreaklineRelief (DataType)

DEFINITION:	ADEOfBreaklineRelief acts as a hook to define properties within an ADE that are to be added to a BreaklineRelief.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.10.5.3. ADEOfMassPointRelief

Table 386 – Metadata of ADEOfMassPointRelief (DataType)

DEFINITION:	ADEOfMassPointRelief acts as a hook to define properties within an ADE that are to be added to a MassPointRelief.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.10.5.4. ADEOfRasterRelief

Table 387 – Metadata of ADEOfRasterRelief (DataType)

DEFINITION:	ADEOfRasterRelief acts as a hook to define properties within an ADE that are to be added to a RasterRelief.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.10.5.5. ADEOfReliefFeature

Table 388 – Metadata of ADEOfReliefFeature (DataType)

DEFINITION:	ADEOfReliefFeature acts as a hook to define properties within an ADE that are to be added to a ReliefFeature.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.10.5.6. ADEOfTINRelief

Table 389 – Metadata of ADEOfTINRelief (DataType)

DEFINITION:	ADEOfTINRelief acts as a hook to define properties within an ADE that are to be added to a TINRelief.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.10.6. Enumerations

None.

8.11. Transportation

Table 390 – Metadata of Transportation (ApplicationSchema)

DESCRIPTION: The Transportation module supports representation of the transportation infrastructure. Transportation features include roads, tracks, waterways, railways, and squares. Transportation features may be represented as a network and/or as a collection of spaces or surface elements embedded in a three-dimensional space.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.11.1. Classes

8.11.1.1. AbstractTransportationSpace

Table 391 – Metadata of AbstractTransportationSpace (FeatureType)

DEFINITION: AbstractTransportationSpace is the abstract superclass of transportation objects such as Roads, Tracks, Railways, Waterways or Squares.

SUBCLASS OF: AbstractUnoccupiedSpace

STEREOTYPE: «FeatureType»

Table 392 – Associations of AbstractTransportationSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
auxiliaryTrafficSpace	AuxiliaryTrafficSpace [1..*]	Relates to the auxiliary traffic spaces that are part of the transportation space.
hole	Hole [1..*]	Relates to the holes that are part of the transportation space.
trafficSpace	TrafficSpace [1..*]	Relates to the traffic spaces that are part of the transportation space.
marking	Marking [1..*]	Relates to the markings that are part of the transportation space.

Table 393 – Attributes of AbstractTransportationSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractTransportationSpace	ADEOfAbstractTransportationSpace [0..*]	Augments AbstractTransportationSpace with properties defined in an ADE.
occupancy	Occupancy [0..*]	Provides information on the residency of persons, vehicles, or other moving features in the transportation space.
trafficDirection	TrafficDirectionValue [0..1]	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.2. AuxiliaryTrafficArea

Table 394 – Metadata of AuxiliaryTrafficArea (FeatureType)

DEFINITION: An AuxiliaryTrafficArea is the ground surface of an AuxiliaryTrafficSpace.

SUBCLASS OF: AbstractThematicSurface

STEREOTYPE: «FeatureType»

Table 395 – Attributes of AuxiliaryTrafficArea (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAuxiliaryTrafficArea	ADEOfAuxiliaryTrafficArea [0..*]	Augments the AuxiliaryTrafficArea with properties defined in an ADE.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
class	AuxiliaryTrafficArea ClassValue [0..1]	Indicates the specific type of the AuxiliaryTrafficArea.
function	AuxiliaryTrafficArea FunctionValue [0..*]	Specifies the intended purposes of the AuxiliaryTrafficArea.
surfaceMaterial	SurfaceMaterialValue [0..1]	Specifies the type of pavement of the AuxiliaryTrafficArea.
usage	AuxiliaryTrafficArea UsageValue [0..*]	Specifies the actual uses of the AuxiliaryTrafficArea.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.3. AuxiliaryTrafficSpace

Table 396 – Metadata of AuxiliaryTrafficSpace (FeatureType)

DEFINITION:	An AuxiliaryTrafficSpace is a space within the transportation space not intended for traffic purposes.
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 397 – Associations of AuxiliaryTrafficSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AuxiliaryTrafficArea [1..*]	Relates to the auxiliary traffic areas that bound the AuxiliaryTrafficSpace. This relation is inherited from the Core module.

Table 398 – Attributes of AuxiliaryTrafficSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAuxiliaryTrafficSpace	ADEOfAuxiliaryTrafficSpace [0..*]	Augments the AuxiliaryTrafficSpace with properties defined in an ADE.
class	AuxiliaryTrafficSpace ClassValue [0..1]	Indicates the specific type of the AuxiliaryTrafficSpace.
function	AuxiliaryTrafficSpace FunctionValue [0..*]	Specifies the intended purposes of the AuxiliaryTrafficSpace.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
granularity	GranularityValue [1..1]	Defines whether auxiliary traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of spatial and semantic decomposition.
usage	AuxiliaryTrafficSpace UsageValue [0..*]	Specifies the actual uses of the AuxiliaryTrafficSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.4. ClearanceSpace

Table 399 – Metadata of ClearanceSpace (FeatureType)

DEFINITION:	A ClearanceSpace represents the actual free space above a TrafficArea within which a mobile object can move without contacting an obstruction.
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 400 – Attributes of ClearanceSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfClearanceSpace	ADEOfClearanceSpace [0..*]	Augments the ClearanceSpace with properties defined in an ADE.
class	ClearanceSpaceClass Value [0..*]	Indicates the specific type of the ClearanceSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.5. Hole

Table 401 – Metadata of Hole (FeatureType)

DEFINITION:	A Hole is an opening in the surface of a Road, Track or Square such as road damages, manholes or drains. Holes can span multiple transportation objects.
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 402 – Associations of Hole (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematicSurface [1..*]	Relates to the surfaces that bound the Hole. This relation is inherited from the Core module.

Table 403 – Attributes of Hole (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfHole	ADEOfHole [0..*]	Augments the Hole with properties defined in an ADE.
class	HoleClassValue [0..1]	Indicates the specific type of the Hole.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.6. HoleSurface

Table 404 – Metadata of HoleSurface (FeatureType)

DEFINITION: A HoleSurface is a representation of the ground surface of a hole.

SUBCLASS OF: AbstractThematicSurface

STEREOTYPE: «FeatureType»

Table 405 – Attributes of HoleSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfHoleSurface	ADEOfHoleSurface [0..*]	Augments the HoleSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.7. Intersection

Table 406 – Metadata of Intersection (FeatureType)

DEFINITION:	An Intersection is a transportation space that is a shared segment of multiple Road, Track, Railway, or Waterway objects (e.g. a crossing of two roads or a level crossing of a road and a railway).
SUBCLASS OF:	AbstractTransportationSpace
STEREOTYPE:	«FeatureType»

Table 407 – Attributes of Intersection (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfIntersection	ADEOfIntersection [0..*]	Augments the Intersection with properties defined in an ADE.
class	IntersectionClassValue [0..1]	Indicates the specific type of the Intersection.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.8. Marking

Table 408 – Metadata of Marking (FeatureType)

DEFINITION:	A Marking is a visible pattern on a transportation area relevant to the structuring or restriction of traffic. Examples are road markings and markings related to railway or waterway traffic.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«FeatureType»

Table 409 – Attributes of Marking (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfMarking	ADEOfMarking [0..*]	Augments the Marking with properties defined in an ADE.
class	MarkingClassValue [0..1]	Indicates the specific type of the Marking.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.9. Railway

Table 410 – Metadata of Railway (TopLevelFeatureType)

DEFINITION: A Railway is a transportation space used by wheeled vehicles on rails.

SUBCLASS OF: AbstractTransportationSpace

STEREOTYPE: «TopLevelFeatureType»

Table 411 – Associations of Railway (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
intersection	Intersection [1..*]	Relates to the intersections that are part of the Railway.
section	Section [1..*]	Relates to the sections that are part of the Railway.

Table 412 – Attributes of Railway (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfRailway	ADEOfRailway [0..*]	Augments the Railway with properties defined in an ADE.
class	RailwayClassValue [0..1]	Indicates the specific type of the Railway.
function	RailwayFunctionValue [0..*]	Specifies the intended purposes of the Railway.
usage	RailwayUsageValue [0..*]	Specifies the actual uses of the Railway.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.10. Road

Table 413 – Metadata of Road (TopLevelFeatureType)

DEFINITION: A Road is a transportation space used by vehicles, bicycles and/or pedestrians.

SUBCLASS OF: AbstractTransportationSpace

STEREOTYPE: «TopLevelFeatureType»

Table 414 – Associations of Road (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
intersection	Intersection [1..*]	Relates to the intersections that are part of the Road.
section	Section [1..*]	Relates to the sections that are part of the Road.

Table 415 – Attributes of Road (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfRoad	ADEOfRoad [0..*]	Augments the Road with properties defined in an ADE.
class	RoadClassValue [0..1]	Indicates the specific type of the Road.
function	RoadFunctionValue [0..*]	Specifies the intended purposes of the Road.
usage	RoadUsageValue [0..*]	Specifies the actual uses of the Road.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.11. Section

Table 416 – Metadata of Section (FeatureType)

DEFINITION: A Section is a transportation space that is a segment of a Road, Railway, Track, or Waterway.

SUBCLASS OF: AbstractTransportationSpace

STEREOTYPE: «FeatureType»

Table 417 – Attributes of Section (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfSection	ADEOfSection [0..*]	Augments the Section with properties defined in an ADE.
class	SectionClassValue [0..1]	Indicates the specific type of the Section.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.12. Square

Table 418 – Metadata of Square (TopLevelFeatureType)

DEFINITION:	A Square is a transportation space for unrestricted movement for vehicles, bicycles and/or pedestrians. This includes plazas as well as large sealed surfaces such as parking lots.
SUBCLASS OF:	AbstractTransportationSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 419 – Attributes of Square (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfSquare	ADEOfSquare [0..*]	Augments the Square with properties defined in an ADE.
class	SquareClassValue [0..1]	Indicates the specific type of the Square.
function	SquareFunctionValue [0..*]	Specifies the intended purposes of the Square.
usage	SquareUsageValue [0..*]	Specifies the actual uses of the Square.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.13. Track

Table 420 – Metadata of Track (TopLevelFeatureType)

DEFINITION:	A Track is a small path mainly used by pedestrians. Tracks can be segmented into Sections and Intersections.
-------------	--

SUBCLASS OF:	AbstractTransportationSpace
---------------------	-----------------------------

| **STEREOTYPE:** | «TopLevelFeatureType» |

Table 421 – Associations of Track (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
intersection	Intersection [1..*]	Relates to the intersections that are part of the Track.
section	Section [1..*]	Relates to the sections that are part of the Track.

Table 422 – Attributes of Track (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTrack	ADEOfTrack [0..*]	Augments the Track with properties defined in an ADE.
class	TrackClassValue [0..1]	Indicates the specific type of the Track.
function	TrackFunctionValue [0..*]	Specifies the intended purposes of the Track.
usage	TrackUsageValue [0..*]	Specifies the actual uses of the Track.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.14. TrafficArea

Table 423 – Metadata of TrafficArea (FeatureType)

DEFINITION:	A TrafficArea is the ground surface of a TrafficSpace. Traffic areas are the surfaces upon which traffic actually takes place.
--------------------	--

| **SUBCLASS OF:** | AbstractThematicSurface |
| **STEREOTYPE:** | «FeatureType» |

Table 424 – Attributes of TrafficArea (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTrafficArea	ADEOfTrafficArea [0..*]	Augments the TrafficArea with properties defined in an ADE.
class	TrafficAreaClassValue [0..1]	Indicates the specific type of the TrafficArea.
function	TrafficAreaFunctionValue [0..*]	Specifies the intended purposes of the TrafficArea.
surfaceMaterial	SurfaceMaterialValue [0..1]	Specifies the type of pavement of the TrafficArea.
usage	TrafficAreaUsageValue [0..*]	Specifies the actual uses of the TrafficArea.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.15. TrafficSpace

Table 425 – Metadata of TrafficSpace (FeatureType)

DEFINITION:	A TrafficSpace is a space in which traffic takes place. Traffic includes the movement of entities such as trains, vehicles, pedestrians, ships, or other transportation types.
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 426 – Associations of TrafficSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
predecessor	TrafficSpace [1..*]	Indicates the predecessor(s) of the TrafficSpace.
clearanceSpace	ClearanceSpace [1..*]	Relates to the clearance spaces that are part of the TrafficSpace.
boundary	TrafficArea [1..*]	Relates to the traffic areas that bound the TrafficSpace. This relation is inherited from the Core module.
successor	TrafficSpace [1..*]	Indicates the successor(s) of the TrafficSpace.

Table 427 – Attributes of TrafficSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTrafficSpace	ADEOfTrafficSpace [0..*]	Augments the TrafficSpace with properties defined in an ADE.
class	TrafficSpaceClassValue [0..1]	Indicates the specific type of the TrafficSpace.
function	TrafficSpaceFunctionValue [0..*]	Specifies the intended purposes of the TrafficSpace.
granularity	GranularityValue [1..1]	Defines whether traffic spaces are represented by individual ways or by individual lanes, depending on the desired level of spatial and semantic decomposition.
occupancy	Occupancy [0..*]	Provides information on the residency of persons, vehicles, or other moving features in the TrafficSpace.
trafficDirection	TrafficDirectionValue [0..1]	Indicates the direction of traffic flow relative to the corresponding linear geometry representation.
usage	TrafficSpaceUsageValue [0..*]	Specifies the actual uses of the TrafficSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.1.16. Waterway

Table 428 – Metadata of Waterway (TopLevelFeatureType)

DEFINITION:	A Waterway is a transportation space used for the movement of vessels upon or within a water body.
SUBCLASS OF:	AbstractTransportationSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 429 – Associations of Waterway (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
intersection	Intersection [1..*]	Relates to the intersections that are part of the Waterway.
section	Section [1..*]	Relates to the sections that are part of the Waterway.

Table 430 – Attributes of Waterway (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWaterway	ADEOfWaterway [0..*]	Augments the Waterway with properties defined in an ADE.
class	WaterwayClassValue [0..1]	Indicates the specific type of the Waterway.
function	WaterwayFunction Value [0..*]	Specifies the intended purposes of the Waterway.
usage	WaterwayUsageValue [0..*]	Specifies the actual uses of the Waterway.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.11.2. Basic types

None.

8.11.3. Unions

None.

8.11.4. Code lists

8.11.4.1. AuxiliaryTrafficAreaClassValue

Table 431 – Metadata of AuxiliaryTrafficAreaClassValue (CodeList)

DEFINITION:	AuxiliaryTrafficAreaClassValue is a code list used to further classify an AuxiliaryTrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.2. AuxiliaryTrafficAreaFunctionValue

Table 432 – Metadata of AuxiliaryTrafficAreaFunctionValue (CodeList)

DEFINITION:	AuxiliaryTrafficAreaFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.3. AuxiliaryTrafficAreaUsageValue

Table 433 – Metadata of AuxiliaryTrafficAreaUsageValue (CodeList)

DEFINITION:	AuxiliaryTrafficAreaUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.4. AuxiliaryTrafficSpaceClassValue

Table 434 – Metadata of AuxiliaryTrafficSpaceClassValue (CodeList)

DEFINITION:	AuxiliaryTrafficSpaceClassValue is a code list used to further classify an AuxiliaryTraffic Space.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.5. AuxiliaryTrafficSpaceFunctionValue

Table 435 – Metadata of AuxiliaryTrafficSpaceFunctionValue (CodeList)

DEFINITION:	AuxiliaryTrafficSpaceFunctionValue is a code list that enumerates the different purposes of an AuxiliaryTrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.6. AuxiliaryTrafficSpaceUsageValue

Table 436 – Metadata of AuxiliaryTrafficSpaceUsageValue (CodeList)

DEFINITION:	AuxiliaryTrafficSpaceUsageValue is a code list that enumerates the different uses of an AuxiliaryTrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.7. ClearanceSpaceClassValue

Table 437 – Metadata of ClearanceSpaceClassValue (CodeList)

DEFINITION:	ClearanceSpaceClassValue is a code list used to further classify a ClearanceSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.8. HoleClassValue

Table 438 – Metadata of HoleClassValue (CodeList)

DEFINITION:	HoleClassValue is a code list used to further classify a Hole.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.9. IntersectionClassValue

Table 439 – Metadata of IntersectionClassValue (CodeList)

DEFINITION:	IntersectionClassValue is a code list used to further classify an Intersection.
--------------------	---

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.10. MarkingClassValue

Table 440 – Metadata of MarkingClassValue (CodeList)

DEFINITION: MarkingClassValue is a code list used to further classify a Marking.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.11. RailwayClassValue

Table 441 – Metadata of RailwayClassValue (CodeList)

DEFINITION: RailwayClassValue is a code list used to further classify a Railway.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.12. RailwayFunctionValue

Table 442 – Metadata of RailwayFunctionValue (CodeList)

DEFINITION: RailwayFunctionValue is a code list that enumerates the different purposes of a Railway.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.13. RailwayUsageValue

Table 443 – Metadata of RailwayUsageValue (CodeList)

DEFINITION:	RailwayUsageValue is a code list that enumerates the different uses of a Railway.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.14. RoadClassValue

Table 444 – Metadata of RoadClassValue (CodeList)

DEFINITION:	RoadClassValue is a code list used to further classify a Road.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.15. RoadFunctionValue

Table 445 – Metadata of RoadFunctionValue (CodeList)

DEFINITION:	RoadFunctionValue is a code list that enumerates the different purposes of a Road.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.16. RoadUsageValue

Table 446 – Metadata of RoadUsageValue (CodeList)

DEFINITION:	RoadUsageValue is a code list that enumerates the different uses of a Road.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.17. SectionClassValue

Table 447 – Metadata of SectionClassValue (CodeList)

DEFINITION: SectionClassValue is a code list used to further classify a Section.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.18. SquareClassValue

Table 448 – Metadata of SquareClassValue (CodeList)

DEFINITION: SquareClassValue is a code list used to further classify a Square.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.19. SquareFunctionValue

Table 449 – Metadata of SquareFunctionValue (CodeList)

DEFINITION: SquareFunctionValue is a code list that enumerates the different purposes of a Square.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.20. SquareUsageValue

Table 450 – Metadata of SquareUsageValue (CodeList)

DEFINITION: SquareUsageValue is a code list that enumerates the different uses of a Square.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.21. SurfaceMaterialValue

Table 451 – Metadata of SurfaceMaterialValue (CodeList)

DEFINITION: SurfaceMaterialValue is a code list that enumerates the different surface materials.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.22. TrackClassValue

Table 452 – Metadata of TrackClassValue (CodeList)

DEFINITION: TrackClassValue is a code list used to further classify a Track.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.23. TrackFunctionValue

Table 453 – Metadata of TrackFunctionValue (CodeList)

DEFINITION: TrackFunctionValue is a code list that enumerates the different purposes of a Track.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.24. TrackUsageValue

Table 454 – Metadata of TrackUsageValue (CodeList)

DEFINITION:	TrackUsageValue is a code list that enumerates the different uses of a Track.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.25. TrafficAreaClassValue

Table 455 – Metadata of TrafficAreaClassValue (CodeList)

DEFINITION:	TrafficAreaClassValue is a code list used to further classify a TrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.26. TrafficAreaFunctionValue

Table 456 – Metadata of TrafficAreaFunctionValue (CodeList)

DEFINITION:	TrafficAreaFunctionValue is a code list that enumerates the different purposes of a Traffic Area.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.27. TrafficAreaUsageValue

Table 457 – Metadata of TrafficAreaUsageValue (CodeList)

DEFINITION:	TrafficAreaUsageValue is a code list that enumerates the different uses of a TrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.28. TrafficSpaceClassValue

Table 458 – Metadata of TrafficSpaceClassValue (CodeList)

DEFINITION:	TrafficSpaceClassValue is a code list used to further classify a TrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.29. TrafficSpaceFunctionValue

Table 459 – Metadata of TrafficSpaceFunctionValue (CodeList)

DEFINITION:	TrafficSpaceFunctionValue is a code list that enumerates the different purposes of a Traffic Space.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.30. TrafficSpaceUsageValue

Table 460 – Metadata of TrafficSpaceUsageValue (CodeList)

DEFINITION:	TrafficSpaceUsageValue is a code list that enumerates the different uses of a TrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.11.4.31. WaterwayClassValue

Table 461 – Metadata of WaterwayClassValue (CodeList)

DEFINITION:	WaterwayClassValue is a code list used to further classify a Waterway.
-------------	--

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.32. WaterwayFunctionValue

Table 462 – Metadata of WaterwayFunctionValue (CodeList)

DEFINITION: WaterwayFunctionValue is a code list that enumerates the different purposes of a Waterway.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.4.33. WaterwayUsageValue

Table 463 – Metadata of WaterwayUsageValue (CodeList)

DEFINITION: WaterwayUsageValue is a code list that enumerates the different uses of a Waterway.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.11.5. Data types

8.11.5.1. ADEOfAbstractTransportationSpace

Table 464 – Metadata of ADEOfAbstractTransportationSpace (DataType)

DEFINITION: ADEOfAbstractTransportationSpace acts as a hook to define properties within an ADE that are to be added to AbstractTransportationSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.5.2. ADEOfAuxiliaryTrafficArea

Table 465 – Metadata of ADEOfAuxiliaryTrafficArea (DataType)

DEFINITION:	ADEOfAuxiliaryTrafficArea acts as a hook to define properties within an ADE that are to be added to an AuxiliaryTrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.3. ADEOfAuxiliaryTrafficSpace

Table 466 – Metadata of ADEOfAuxiliaryTrafficSpace (DataType)

DEFINITION:	ADEOfAuxiliaryTrafficSpace acts as a hook to define properties within an ADE that are to be added to an AuxiliaryTrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.4. ADEOfClearanceSpace

Table 467 – Metadata of ADEOfClearanceSpace (DataType)

DEFINITION:	ADEOfClearanceSpace acts as a hook to define properties within an ADE that are to be added to a ClearanceSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.5. ADEOfHole

Table 468 – Metadata of ADEOfHole (DataType)

DEFINITION:	ADEOfHole acts as a hook to define properties within an ADE that are to be added to a Hole.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.5.6. ADEOfHoleSurface

Table 469 – Metadata of ADEOfHoleSurface (DataType)

DEFINITION: ADEOfHoleSurface acts as a hook to define properties within an ADE that are to be added to a HoleSurface.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.5.7. ADEOfIntersection

Table 470 – Metadata of ADEOfIntersection (DataType)

DEFINITION: ADEOfIntersection acts as a hook to define properties within an ADE that are to be added to an Intersection.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.5.8. ADEOfMarking

Table 471 – Metadata of ADEOfMarking (DataType)

DEFINITION: ADEOfMarking acts as a hook to define properties within an ADE that are to be added to a Marking.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.5.9. ADEOfRailway

Table 472 – Metadata of ADEOfRailway (DataType)

DEFINITION:	ADEOfRailway acts as a hook to define properties within an ADE that are to be added to a Railway.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.10. ADEOfRoad

Table 473 – Metadata of ADEOfRoad (DataType)

DEFINITION:	ADEOfRoad acts as a hook to define properties within an ADE that are to be added to a Road.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.11. ADEOfSection

Table 474 – Metadata of ADEOfSection (DataType)

DEFINITION:	ADEOfSection acts as a hook to define properties within an ADE that are to be added to a Section.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.12. ADEOfSquare

Table 475 – Metadata of ADEOfSquare (DataType)

DEFINITION:	ADEOfSquare acts as a hook to define properties within an ADE that are to be added to a Square.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.13. ADEOfTrack

Table 476 – Metadata of ADEOfTrack (DataType)

DEFINITION:	ADEOfTrack acts as a hook to define properties within an ADE that are to be added to a Track.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.14. ADEOfTrafficArea

Table 477 – Metadata of ADEOfTrafficArea (DataType)

DEFINITION:	ADEOfTrafficArea acts as a hook to define properties within an ADE that are to be added to a TrafficArea.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.15. ADEOfTrafficSpace

Table 478 – Metadata of ADEOfTrafficSpace (DataType)

DEFINITION:	ADEOfTrafficSpace acts as a hook to define properties within an ADE that are to be added to a TrafficSpace.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.11.5.16. ADEOfWaterway

Table 479 – Metadata of ADEOfWaterway (DataType)

DEFINITION:	ADEOfWaterway acts as a hook to define properties within an ADE that are to be added to a Waterway.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.11.6. Enumerations

8.11.6.1. GranularityValue

Table 480 – Metadata of GranularityValue (Enumeration)

DEFINITION: GranularityValue enumerates the different levels of granularity in which transportation objects are represented.

STEREOTYPE: «Enumeration»

Table 481 – Values of GranularityValue (Enumeration)

LITERAL VALUE	DEFINITION
lane	Indicates that the individual lanes of the transportation object are represented.
way	Indicates that the individual (carriage)ways of the transportation object are represented.

8.11.6.2. TrafficDirectionValue

Table 482 – Metadata of TrafficDirectionValue (Enumeration)

DEFINITION: TrafficDirectionValue enumerates the allowed directions of travel of a mobile object.

STEREOTYPE: «Enumeration»

Table 483 – Values of TrafficDirectionValue (Enumeration)

LITERAL VALUE	DEFINITION
forwards	Indicates that traffic flows in the direction of the corresponding linear geometry.
backwards	Indicates that traffic flows in the opposite direction of the corresponding linear geometry.

LITERAL VALUE	DEFINITION
both	Indicates that traffic flows in both directions.

8.12. Vegetation

Table 484 – Metadata of Vegetation (ApplicationSchema)

DESCRIPTION:	The Vegetation module supports representation of vegetation objects with vegetation-specific thematic classes. CityGML's vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.12.1. Classes

8.12.1.1. AbstractVegetationObject

Table 485 – Metadata of AbstractVegetationObject (FeatureType)

DEFINITION:	AbstractVegetationObject is the abstract superclass for all kinds of vegetation objects.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«FeatureType»

Table 486 – Attributes of AbstractVegetationObject (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractVegetationObject	ADEOfAbstractVegetationObject [0..*]	Augments AbstractVegetationObject with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.12.1.2. PlantCover

Table 487 – Metadata of PlantCover (TopLevelFeatureType)

DEFINITION: A PlantCover represents a space covered by vegetation.

SUBCLASS OF:	AbstractVegetationObject
---------------------	--------------------------

STEREOTYPE:	«TopLevelFeatureType»
--------------------	-----------------------

Table 488 – Attributes of PlantCover (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfPlantCover	ADEOfPlantCover [0..*]	Augments the PlantCover with properties defined in an ADE.
averageHeight	Length [0..1]	Specifies the average height of the PlantCover.
class	PlantCoverClassValue [0..1]	Indicates the specific type of the PlantCover.
function	PlantCoverFunctionValue [0..*]	Specifies the intended purposes of the PlantCover.
maxHeight	Length [0..1]	Specifies the maximum height of the PlantCover.
minHeight	Length [0..1]	Specifies the minimum height of the PlantCover.
usage	PlantCoverUsageValue [0..*]	Specifies the actual uses of the PlantCover.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.12.1.3. SolitaryVegetationObject

Table 489 – Metadata of SolitaryVegetationObject (TopLevelFeatureType)

DEFINITION: A SolitaryVegetationObject represents individual vegetation objects, e.g. trees or bushes.

SUBCLASS OF:	AbstractVegetationObject
---------------------	--------------------------

STEREOTYPE:	«TopLevelFeatureType»
--------------------	-----------------------

Table 490 – Attributes of SolitaryVegetationObject (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfSolitaryVegetationObject	ADEOfSolitaryVegetationObject [0..*]	Augments the SolitaryVegetationObject with properties defined in an ADE.
class	SolitaryVegetationObjectClassValue [0..1]	Indicates the specific type of the SolitaryVegetationObject.
crownDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's crown.
function	SolitaryVegetationObjectFunctionValue [0..*]	Specifies the intended purposes of the SolitaryVegetationObject.
height	Length [0..1]	Distance between the highest point of the vegetation object and the lowest point of the terrain at the bottom of the object.
maxRootBallDepth	Length [0..1]	Specifies the vertical distance between the lowest point of the SolitaryVegetationObject's root ball and the terrain surface.
rootBallDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's root ball.
species	SpeciesValue [0..1]	Indicates the botanical name of the SolitaryVegetationObject.
trunkDiameter	Length [0..1]	Specifies the diameter of the SolitaryCityObject's trunk.
usage	SolitaryVegetationObjectUsageValue [0..*]	Specifies the actual uses of the SolitaryVegetationObject.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.12.2. Basic types

None.

8.12.3. Unions

None.

8.12.4. Code lists

8.12.4.1. PlantCoverClassValue

Table 491 – Metadata of PlantCoverClassValue (CodeList)

DEFINITION: PlantCoverClassValue is a code list used to further classify a PlantCover.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.12.4.2. PlantCoverFunctionValue

Table 492 – Metadata of PlantCoverFunctionValue (CodeList)

DEFINITION: PlantCoverFunctionValue is a code list that enumerates the different purposes of a Plant Cover.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.12.4.3. PlantCoverUsageValue

Table 493 – Metadata of PlantCoverUsageValue (CodeList)

DEFINITION: PlantCoverUsageValue is a code list that enumerates the different uses of a PlantCover.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.12.4.4. SolitaryVegetationObjectClassValue

Table 494 – Metadata of SolitaryVegetationObjectClassValue (CodeList)

DEFINITION:	SolitaryVegetationObjectClassValue is a code list used to further classify a SolitaryVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.12.4.5. SolitaryVegetationObjectFunctionValue

Table 495 – Metadata of SolitaryVegetationObjectFunctionValue (CodeList)

DEFINITION:	SolitaryVegetationObjectFunctionValue is a code list that enumerates the different purposes of a SolitaryVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.12.4.6. SolitaryVegetationObjectUsageValue

Table 496 – Metadata of SolitaryVegetationObjectUsageValue (CodeList)

DEFINITION:	SolitaryVegetationObjectUsageValue is a code list that enumerates the different uses of a SolitaryVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.12.4.7. SpeciesValue

Table 497 – Metadata of SpeciesValue (CodeList)

DEFINITION:	A SpeciesValue is a code list that enumerates the species of a SolitaryVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.12.5. Data types

8.12.5.1. ADEOfAbstractVegetationObject

Table 498 – Metadata of ADEOfAbstractVegetationObject (DataType)

DEFINITION:	ADEOfAbstractVegetationObject acts as a hook to define properties within an ADE that are to be added to AbstractVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.12.5.2. ADEOfPlantCover

Table 499 – Metadata of ADEOfPlantCover (DataType)

DEFINITION:	ADEOfPlantCover acts as a hook to define properties within an ADE that are to be added to a PlantCover.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.12.5.3. ADEOfSolitaryVegetationObject

Table 500 – Metadata of ADEOfSolitaryVegetationObject (DataType)

DEFINITION:	ADEOfSolitaryVegetationObject acts as a hook to define properties within an ADE that are to be added to a SolitaryVegetationObject.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.12.6. Enumerations

None.

8.13. Versioning

Table 501 – Metadata of Versioning (ApplicationSchema)

DESCRIPTION: The Versioning module supports representation of multiple versions of CityGML features within a single CityGML model. In addition, also the version transitions and transactions that lead to the different versions can be represented.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.13.1. Classes

8.13.1.1. Version

Table 502 – Metadata of Version (FeatureType)

DEFINITION: Version represents a defined state of a city model consisting of the dedicated versions of all city object instances that belong to the respective city model version. Versions can have names, a description and can be labeled with an arbitrary number of user defined tags.

SUBCLASS OF: AbstractVersion

STEREOTYPE: «FeatureType»

Table 503 – Associations of Version (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
versionMember	AbstractFeatureWithLifespan [1..*]	Relates to all city objects that are part of the city model version.

Table 504 – Attributes of Version (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfVersion	ADEOfVersion [0..*]	Augments the Version with properties defined in an ADE.
tag	CharacterString [0..*]	Allows for adding keywords to the city model version.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.13.1.2. VersionTransition

Table 505 – Metadata of VersionTransition (FeatureType)

DEFINITION:	VersionTransition describes the change of the state of a city model from one version to another. Version transitions can have names, a description and can be further qualified by a type and a reason.
SUBCLASS OF:	AbstractVersionTransition
STEREOTYPE:	«FeatureType»

Table 506 – Associations of VersionTransition (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
from	Version [0..1]	Relates to the predecessor version of the VersionTransition.
to	Version [0..1]	Relates to the successor version of the VersionTransition.
transaction	Transaction [1..*]	Relates to all transactions that have been applied as part of the Version Transition.

Table 507 – Attributes of VersionTransition (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfVersionTransition	ADEOfVersionTransition [0..*]	Augments the VersionTransition with properties defined in an ADE.
clonePredecessor	Boolean [1..1]	Indicates whether the set of city object instances belonging to the successor version of the city model is either explicitly enumerated within the successor version object (attribute clonePredecessor=false), or has to be derived from the modifications of the city model provided as a list of transactions on the city object versions contained in the predecessor version (attribute clonePredecessor=true).
reason	CharacterString [0..1]	Specifies why the VersionTransition has been carried out.
type	TransitionTypeValue [0..1]	Indicates the specific type of the VersionTransition.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.13.2. Basic types

None.

8.13.3. Unions

None.

8.13.4. Code lists

None.

8.13.5. Data types

8.13.5.1. ADEOfVersion

Table 508 – Metadata of ADEOfVersion (DataType)

DEFINITION:	ADEOfVersion acts as a hook to define properties within an ADE that are to be added to a Version.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.13.5.2. ADEOfVersionTransition

Table 509 – Metadata of ADEOfVersionTransition (DataType)

DEFINITION:	ADEOfVersionTransition acts as a hook to define properties within an ADE that are to be added to a VersionTransition.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.13.5.3. Transaction

Table 510 – Metadata of Transaction (DataType)

DEFINITION:	Transaction represents a modification of the city model by the creation, termination, or replacement of a specific city object. While the creation of a city object also marks its first object version, the termination marks the end of existence of a real world object and, hence, also terminates the final version of a city object. The replacement of a city object means that a specific version of it is replaced by a new version.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 511 – Associations of Transaction (DataType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
newFeature	AbstractFeatureWith Lifespan [0..1]	Relates to the version of the city object subsequent to the Transaction.
oldFeature	AbstractFeatureWith Lifespan [0..1]	Relates to the version of the city object prior to the Transaction.

Table 512 – Attributes of Transaction (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
type	TransactionTypeValue [1..1]	Indicates the specific type of the Transaction.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.13.6. Enumerations

8.13.6.1. TransactionTypeValue

Table 513 – Metadata of TransactionTypeValue (Enumeration)

DEFINITION:	TransactionTypeValue enumerates the three possible types of transactions: insert, delete, or replace.
--------------------	---

STEREOTYPE:	«Enumeration»
--------------------	---------------

Table 514 – Values of TransactionTypeValue (Enumeration)

LITERAL VALUE	DEFINITION
insert	Indicates that the feature referenced from the Transaction via the “newFeature” association has been newly created; the association “oldFeature” is empty in this case.
delete	Indicates that the feature referenced from the Transaction via the “oldFeature” association ceases to exist; the association “newFeature” is empty in this case.
replace	Indicates that the feature referenced from the Transaction via the “oldFeature” association has been replaced by the feature referenced via the “newFeature” association.

8.13.6.2. TransitionTypeValue

Table 515 – Metadata of TransitionTypeValue (Enumeration)

DEFINITION:	TransitionTypeValue enumerates the different kinds of version transitions. “planned” and “fork” should be used in cases when from one city model version multiple successor versions are being created. “realized” and “merge” should be used when different city model versions are converging into a common successor version.
STEREOTYPE:	«Enumeration»

Table 516 – Values of TransitionTypeValue (Enumeration)

LITERAL VALUE	DEFINITION
planned	Indicates that the successor version of the city model represents a planning state for a possible future of the city.
realized	Indicates that the predecessor version is the chosen one from a number of possible planning versions.
historicalSuccession	Indicates that the successor version reflects updates on the city model over time (historical timeline). It shall only be used for at most one version transition outgoing from a city model version.
fork	Indicates other reasons to create alternative city model versions, for example, when different parties are updating parts of the city model or to reflect the results of different simulation runs.
merge	Indicates other reasons to converge multiple versions back into a common city model version.

8.14. WaterBody

Table 517 – Metadata of WaterBody (ApplicationSchema)

DESCRIPTION: The WaterBody module supports representation of the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects of fluid flow.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.14.1. Classes

8.14.1.1. AbstractWaterBoundarySurface

Table 518 – Metadata of AbstractWaterBoundarySurface (FeatureType)

DEFINITION: AbstractWaterBoundarySurface is the abstract superclass for all kinds of thematic surfaces bounding a water body.

SUBCLASS OF: AbstractThematicSurface

STEREOTYPE: «FeatureType»

Table 519 – Attributes of AbstractWaterBoundarySurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract WaterBoundary Surface	ADEOfAbstractWater BoundarySurface [0..*]	Augments AbstractWaterBoundarySurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.14.1.2. WaterBody

Table 520 – Metadata of WaterBody (TopLevelFeatureType)

DEFINITION:	A WaterBody represents significant and permanent or semi-permanent accumulations of surface water, usually covering a part of the Earth.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«TopLevelFeatureType»

Table 521 – Associations of WaterBody (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractWater BoundarySurface [1..*]	

Table 522 – Attributes of WaterBody (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWaterBody	ADEOfWaterBody [0..*]	Augments the WaterBody with properties defined in an ADE.
class	WaterBodyClassValue [0..1]	Indicates the specific type of the WaterBody.
function	WaterBodyFunctionValue [0..*]	Specifies the intended purposes of the WaterBody.
usage	WaterBodyUsageValue [0..*]	Specifies the actual uses of the WaterBody.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.14.1.3. WaterGroundSurface

Table 523 – Metadata of WaterGroundSurface (FeatureType)

DEFINITION:	A WaterGroundSurface represents the exterior boundary surface of the submerged bottom of a water body.
SUBCLASS OF:	AbstractWaterBoundarySurface
STEREOTYPE:	«FeatureType»

Table 524 – Attributes of WaterGroundSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWaterGroundSurface	ADEOfWaterGroundSurface [0..*]	Augments the WaterGroundSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.14.1.4. WaterSurface

Table 525 – Metadata of WaterSurface (FeatureType)

DEFINITION:	A WaterSurface represents the upper exterior interface between a water body and the atmosphere.
SUBCLASS OF:	AbstractWaterBoundarySurface
STEREOTYPE:	«FeatureType»

Table 526 – Attributes of WaterSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWaterSurface	ADEOfWaterSurface [0..*]	Augments the WaterSurface with properties defined in an ADE.
waterLevel	WaterLevelValue [0..1]	Specifies the level of the WaterSurface.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.14.2. Basic types

None.

8.14.3. Unions

None.

8.14.4. Code lists

8.14.4.1. WaterBodyClassValue

Table 527 – Metadata of WaterBodyClassValue (CodeList)

DEFINITION: WaterBodyClassValue is a code list used to further classify a WaterBody.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.14.4.2. WaterBodyFunctionValue

Table 528 – Metadata of WaterBodyFunctionValue (CodeList)

DEFINITION: WaterBodyFunctionValue is a code list that enumerates the different purposes of a WaterBody.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.14.4.3. WaterBodyUsageValue

Table 529 – Metadata of WaterBodyUsageValue (CodeList)

DEFINITION: WaterBodyUsageValue is a code list that enumerates the different uses of a WaterBody.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.14.4.4. WaterLevelValue

Table 530 – Metadata of WaterLevelValue (CodeList)

DEFINITION:	WaterLevelValue is a code list that enumerates the different levels of a water surface.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.14.5. Data types

8.14.5.1. ADEOfAbstractWaterBoundarySurface

Table 531 – Metadata of ADEOfAbstractWaterBoundarySurface (DataType)

DEFINITION:	ADEOfAbstractWaterBoundarySurface acts as a hook to define properties within an ADE that are to be added to AbstractWaterBoundarySurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.14.5.2. ADEOfWaterBody

Table 532 – Metadata of ADEOfWaterBody (DataType)

DEFINITION:	ADEOfWaterBody acts as a hook to define properties within an ADE that are to be added to a WaterBody.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.14.5.3. ADEOfWaterGroundSurface

Table 533 – Metadata of ADEOfWaterGroundSurface (DataType)

DEFINITION:	ADEOfWaterGroundSurface acts as a hook to define properties within an ADE that are to be added to a WaterGroundSurface.
SUBCLASS OF:	None

STEREOTYPE:	«DataType»
-------------	------------

8.14.5.4. ADEOfWaterSurface

Table 534 – Metadata of ADEOfWaterSurface (DataType)

DEFINITION:	ADEOfWaterSurface acts as a hook to define properties within an ADE that are to be added to a WaterSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.14.6. Enumerations

None.

8.15. Construction

Table 535 – Metadata of Construction (ApplicationSchema)

DESCRIPTION:	The Construction module supports representation of key elements of different types of constructions. These key elements include construction surfaces (e.g. floor and ceiling), windows and doors, constructive elements (e.g. beams and slabs), installations, and furniture.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.15.1. Classes

8.15.1.1. AbstractConstruction

Table 536 – Metadata of AbstractConstruction (FeatureType)

DEFINITION:	AbstractConstruction is the abstract superclass for objects that are manufactured by humans from construction materials, are connected to earth, and are intended to be permanent. A
-------------	--

connection with the ground also exists when the construction rests by its own weight on the ground or is moveable limited on stationary rails or if the construction is intended to be used mainly stationary.

SUBCLASS OF: AbstractOccupiedSpace

STEREOTYPE: «FeatureType»

Table 537 – Associations of AbstractConstruction (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the construction. This relation is inherited from the Core module.

Table 538 – Attributes of AbstractConstruction (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractConstruction	ADEOfAbstractConstruction [0..*]	Augments AbstractConstruction with properties defined in an ADE.
conditionOfConstruction	ConditionOfConstructionValue [0..1]	Indicates the life-cycle status of the construction. [cf. INSPIRE]
constructionEvent	ConstructionEvent [0..*]	Describes specific events in the life-time of the construction.
dateOfConstruction	Date [0..1]	Indicates the date at which the construction was completed.
dateOfDemolition	Date [0..1]	Indicates the date at which the construction was demolished.
elevation	Elevation [0..*]	Specifies qualified elevations of the construction in relation to a well-defined surface which is commonly taken as origin (e.g. geoid or water level). [cf. INSPIRE]
height	Height [0..*]	Specifies qualified heights of the construction above ground or below ground. [cf. INSPIRE]
occupancy	Occupancy [0..*]	Provides qualified information on the residency of persons, animals, or other moveable objects in the construction.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.2. AbstractConstructionSurface

Table 539 – Metadata of AbstractConstructionSurface (FeatureType)

DEFINITION:	AbstractConstructionSurface is the abstract superclass for different kinds of surfaces that bound a construction.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«FeatureType»

Table 540 – Associations of AbstractConstructionSurface (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
fillingSurface	AbstractFillingSurface [1..*]	Relates to the surfaces that seal the openings of the construction surface.

Table 541 – Attributes of AbstractConstructionSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractConstructionSurface	ADEOfAbstractConstructionSurface [0..*]	Augments AbstractConstructionSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.3. AbstractConstructiveElement

Table 542 – Metadata of AbstractConstructiveElement (FeatureType)

DEFINITION:	AbstractConstructiveElement is the abstract superclass for the representation of volumetric elements of a construction. Examples are walls, beams, slabs.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«FeatureType»

Table 543 – Associations of AbstractConstructiveElement (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
filling	AbstractFillingElement [1..*]	Relates to the elements that fill the opening of the constructive element.

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the constructive element. This relation is inherited from the Core module.

Table 544 – Attributes of AbstractConstructiveElement (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Constructive Element	ADEOfAbstract ConstructiveElement [0..*]	Augments AbstractConstructiveElement with properties defined in an ADE.
isStructural Element	Boolean [0..1]	Indicates whether the constructive element is essential from a structural point of view. A structural element cannot be omitted without collapsing of the construction. Examples are pylons and anchorages of bridges.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.4. AbstractFillingElement

Table 545 – Metadata of AbstractFillingElement (FeatureType)

DEFINITION:	AbstractFillingElement is the abstract superclass for different kinds of elements that fill the openings of a construction.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«FeatureType»

Table 546 – Attributes of AbstractFillingElement (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract FillingElement	ADEOfAbstractFilling Element [0..*]	Augments AbstractFillingElement with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.5. AbstractFillingSurface

Table 547 – Metadata of AbstractFillingSurface (FeatureType)

DEFINITION:	AbstractFillingSurface is the abstract superclass for different kinds of surfaces that seal openings filled by filling elements.
SUBCLASS OF:	AbstractThematicSurface
STEREOTYPE:	«FeatureType»

Table 548 – Attributes of AbstractFillingSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractFillingSurface	ADEOfAbstractFillingSurface [0..*]	Augments AbstractFillingSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.6. AbstractFurniture

Table 549 – Metadata of AbstractFurniture (FeatureType)

DEFINITION:	AbstractFurniture is the abstract superclass for the representation of furniture objects of a construction.
SUBCLASS OF:	AbstractOccupiedSpace
STEREOTYPE:	«FeatureType»

Table 550 – Attributes of AbstractFurniture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractFurniture	ADEOfAbstractFurniture [0..*]	Augments AbstractFurniture with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.7. AbstractInstallation

Table 551 – Metadata of AbstractInstallation (FeatureType)

DEFINITION:	AbstractInstallation is the abstract superclass for the representation of installation objects of a construction.
--------------------	---

SUBCLASS OF: AbstractOccupiedSpace

STEREOTYPE: «FeatureType»

Table 552 – Associations of AbstractInstallation (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the installation. This relation is inherited from the Core module.

Table 553 – Attributes of AbstractInstallation (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Installation	ADEOfAbstract Installation [0..*]	Augments AbstractInstallation with properties defined in an ADE.
relation ToConstruction	Relation ToConstruction [0..1]	Indicates whether the installation is located inside and/or outside of the construction.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.8. CeilingSurface

Table 554 – Metadata of CeilingSurface (FeatureType)

DEFINITION: A CeilingSurface is a surface that represents the interior ceiling of a construction. An example is the ceiling of a room.

SUBCLASS OF: AbstractConstructionSurface

STEREOTYPE: «FeatureType»

Table 555 – Attributes of CeilingSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfCeiling Surface	ADEOfCeilingSurface [0..*]	Augments the CeilingSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.9. Door

Table 556 – Metadata of Door (FeatureType)

DEFINITION:	A Door is a construction for closing an opening intended primarily for access or egress or both. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractFillingElement
STEREOTYPE:	«FeatureType»

Table 557 – Associations of Door (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	DoorSurface [1..*]	Relates to the door surfaces that bound the Door. This relation is inherited from the Core module.
address	Address [1..*]	Relates to the addresses that are assigned to the Door.

Table 558 – Attributes of Door (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfDoor	ADEOfDoor [0..*]	Augments the Door with properties defined in an ADE.
class	DoorClassValue [0..1]	Indicates the specific type of the Door.
function	DoorFunctionValue [0..*]	Specifies the intended purposes of the Door.
usage	DoorUsageValue [0..*]	Specifies the actual uses of the Door.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.10. DoorSurface

Table 559 – Metadata of DoorSurface (FeatureType)

DEFINITION:	A DoorSurface is either a boundary surface of a Door feature or a surface that seals an opening filled by a door.
-------------	---

SUBCLASS OF: AbstractFillingSurface

STEREOTYPE: «FeatureType»

Table 560 – Associations of DoorSurface (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
address	Address [1..*]	Relates to the addresses that are assigned to the DoorSurface.

Table 561 – Attributes of DoorSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfDoorSurface	ADEOfDoorSurface [0..*]	Augments the DoorSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.11. FloorSurface

Table 562 – Metadata of FloorSurface (FeatureType)

DEFINITION: A FloorSurface is surface that represents the interior floor of a construction. An example is the floor of a room.

SUBCLASS OF: AbstractConstructionSurface

STEREOTYPE: «FeatureType»

Table 563 – Attributes of FloorSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfFloorSurface	ADEOfFloorSurface [0..*]	Augments the FloorSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.12. GroundSurface

Table 564 – Metadata of GroundSurface (FeatureType)

DEFINITION:	A GroundSurface is a surface that represents the ground plate of a construction. The polygon defining the ground plate is congruent with the footprint of the construction.
SUBCLASS OF:	AbstractConstructionSurface
STEREOTYPE:	«FeatureType»

Table 565 – Attributes of GroundSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfGroundSurface	ADEOfGroundSurface [0..*]	Augments the GroundSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.13. InteriorWallSurface

Table 566 – Metadata of InteriorWallSurface (FeatureType)

DEFINITION:	An InteriorWallSurface is a surface that is visible from inside a construction. An example is the wall of a room.
SUBCLASS OF:	AbstractConstructionSurface
STEREOTYPE:	«FeatureType»

Table 567 – Attributes of InteriorWallSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfInteriorWallSurface	ADEOfInteriorWallSurface [0..*]	Augments the InteriorWallSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.14. OtherConstruction

Table 568 – Metadata of OtherConstruction (TopLevelFeatureType)

DEFINITION:	An OtherConstruction is a construction that is not covered by any of the other subclasses of AbstractConstruction.
--------------------	--

SUBCLASS OF: AbstractConstruction

STEREOTYPE: «TopLevelFeatureType»

Table 569 – Attributes of OtherConstruction (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfOtherConstruction	ADEOfOtherConstruction [0..*]	Augments the OtherConstruction with properties defined in an ADE.
class	OtherConstructionClassValue [0..1]	Indicates the specific type of the OtherConstruction.
function	OtherConstructionFunctionValue [0..*]	Specifies the intended purposes of the OtherConstruction.
usage	OtherConstructionUsageValue [0..*]	Specifies the actual uses of the OtherConstruction.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.15. OuterCeilingSurface

Table 570 – Metadata of OuterCeilingSurface (FeatureType)

DEFINITION: An OuterCeilingSurface is a surface that belongs to the outer building shell with the orientation pointing downwards. An example is the ceiling of a loggia.

SUBCLASS OF: AbstractConstructionSurface

STEREOTYPE: «FeatureType»

Table 571 – Attributes of OuterCeilingSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfOuterCeilingSurface	ADEOfOuterCeilingSurface [0..*]	Augments the OuterCeilingSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.16. OuterFloorSurface

Table 572 – Metadata of OuterFloorSurface (FeatureType)

DEFINITION:	An OuterFloorSurface is a surface that belongs to the outer construction shell with the orientation pointing upwards. An example is the floor of a loggia.
SUBCLASS OF:	AbstractConstructionSurface
STEREOTYPE:	«FeatureType»

Table 573 – Attributes of OuterFloorSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfOuterFloorSurface	ADEOfOuterFloorSurface [0..*]	Augments the OuterFloorSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.17. RoofSurface

Table 574 – Metadata of RoofSurface (FeatureType)

DEFINITION:	A RoofSurface is a surface that delimits major roof parts of a construction.
SUBCLASS OF:	AbstractConstructionSurface
STEREOTYPE:	«FeatureType»

Table 575 – Attributes of RoofSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfRoofSurface	ADEOfRoofSurface [0..*]	Augments the RoofSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.18. WallSurface

Table 576 – Metadata of WallSurface (FeatureType)

DEFINITION:	A WallSurface is a surface that is part of the building facade visible from the outside.
--------------------	--

SUBCLASS OF: AbstractConstructionSurface

STEREOTYPE: «FeatureType»

Table 577 – Attributes of WallSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWallSurface	ADEOfWallSurface [0..*]	Augments the WallSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.19. Window

Table 578 – Metadata of Window (FeatureType)

DEFINITION:	A Window is a construction for closing an opening in a wall or roof, primarily intended to admit light and/or provide ventilation. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractFillingElement
STEREOTYPE:	«FeatureType»

Table 579 – Associations of Window (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	WindowSurface [1..*]	Relates to the window surfaces that bound the Window. This relation is inherited from the Core module.

Table 580 – Attributes of Window (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWindow	ADEOfWindow [0..*]	Augments the Window with properties defined in an ADE.
class	WindowClassValue [0..1]	Indicates the specific type of the Window.
function	WindowFunctionValue [0..*]	Specifies the intended purposes of the Window.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
usage	WindowUsageValue [0..*]	Specifies the actual uses of the Window.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.1.20. WindowSurface

Table 581 – Metadata of WindowSurface (FeatureType)

DEFINITION:	A WindowSurface is either a boundary surface of a Window feature or a surface that seals an opening filled by a window.
SUBCLASS OF:	AbstractFillingSurface
STEREOTYPE:	«FeatureType»

Table 582 – Attributes of WindowSurface (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfWindowSurface	ADEOfWindowSurface [0..*]	Augments the WindowSurface with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.2. Basic types

None.

8.15.3. Unions

None.

8.15.4. Code lists

8.15.4.1. DoorClassValue

Table 583 – Metadata of DoorClassValue (CodeList)

DEFINITION: DoorClassValue is a code list used to further classify a Door.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.2. DoorFunctionValue

Table 584 – Metadata of DoorFunctionValue (CodeList)

DEFINITION: DoorFunctionValue is a code list that enumerates the different purposes of a Door.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.3. DoorUsageValue

Table 585 – Metadata of DoorUsageValue (CodeList)

DEFINITION: DoorUsageValue is a code list that enumerates the different uses of a Door.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.4. ElevationReferenceValue

Table 586 – Metadata of ElevationReferenceValue (CodeList)

DEFINITION: ElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure construction heights.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.5. EventValue

Table 587 – Metadata of EventValue (CodeList)

DEFINITION:	EventValue is a code list that enumerates the different events of a construction.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.15.4.6. OtherConstructionClassValue

Table 588 – Metadata of OtherConstructionClassValue (CodeList)

DEFINITION:	OtherConstructionClassValue is a code list used to further classify an OtherConstruction.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.15.4.7. OtherConstructionFunctionValue

Table 589 – Metadata of OtherConstructionFunctionValue (CodeList)

DEFINITION:	OtherConstructionFunctionValue is a code list that enumerates the different purposes of an OtherConstruction.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.15.4.8. OtherConstructionUsageValue

Table 590 – Metadata of OtherConstructionUsageValue (CodeList)

DEFINITION:	OtherConstructionUsageValue is a code list that enumerates the different uses of an OtherConstruction.
-------------	--

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.9. WindowClassName

Table 591 – Metadata of WindowClassName (CodeList)

DEFINITION: WindowClassName is a code list used to further classify a Window.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.10. WindowFunctionValue

Table 592 – Metadata of WindowFunctionValue (CodeList)

DEFINITION: WindowFunctionValue is a code list that enumerates the different purposes of a Window.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.4.11. WindowUsageValue

Table 593 – Metadata of WindowUsageValue (CodeList)

DEFINITION: WindowUsageValue is a code list that enumerates the different uses of a Window.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.15.5. Data types

8.15.5.1. ADEOfAbstractConstruction

Table 594 – Metadata of ADEOfAbstractConstruction (DataType)

DEFINITION:	ADEOfAbstractConstruction acts as a hook to define properties within an ADE that are to be added to AbstractConstruction.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.2. ADEOfAbstractConstructionSurface

Table 595 – Metadata of ADEOfAbstractConstructionSurface (DataType)

DEFINITION:	ADEOfAbstractConstructionSurface acts as a hook to define properties within an ADE that are to be added to AbstractConstructionSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.3. ADEOfAbstractConstructiveElement

Table 596 – Metadata of ADEOfAbstractConstructiveElement (DataType)

DEFINITION:	ADEOfAbstractConstructiveElement acts as a hook to define properties within an ADE that are to be added to AbstractConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.4. ADEOfAbstractFillingElement

Table 597 – Metadata of ADEOfAbstractFillingElement (DataType)

DEFINITION:	ADEOfAbstractFillingElement acts as a hook to define properties within an ADE that are to be added to AbstractFillingElement.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.5. ADEOfAbstractFillingSurface

Table 598 – Metadata of ADEOfAbstractFillingSurface (DataType)

DEFINITION:	ADEOfAbstractFillingSurface acts as a hook to define properties within an ADE that are to be added to AbstractFillingSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.6. ADEOfAbstractFurniture

Table 599 – Metadata of ADEOfAbstractFurniture (DataType)

DEFINITION:	ADEOfAbstractFurniture acts as a hook to define properties within an ADE that are to be added to AbstractFurniture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.7. ADEOfAbstractInstallation

Table 600 – Metadata of ADEOfAbstractInstallation (DataType)

DEFINITION:	ADEOfAbstractInstallation acts as a hook to define properties within an ADE that are to be added to AbstractInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.8. ADEOfCeilingSurface

Table 601 – Metadata of ADEOfCeilingSurface (DataType)

DEFINITION:	ADEOfCeilingSurface acts as a hook to define properties within an ADE that are to be added to a CeilingSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.9. ADEOfDoor

Table 602 – Metadata of ADEOfDoor (DataType)

DEFINITION:	ADEOfDoor acts as a hook to define properties within an ADE that are to be added to a Door.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.10. ADEOfDoorSurface

Table 603 – Metadata of ADEOfDoorSurface (DataType)

DEFINITION:	ADEOfDoorSurface acts as a hook to define properties within an ADE that are to be added to a DoorSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.11. ADEOfFloorSurface

Table 604 – Metadata of ADEOfFloorSurface (DataType)

DEFINITION:	ADEOfFloorSurface acts as a hook to define properties within an ADE that are to be added to a FloorSurface.
-------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.15.5.12. ADEOfGroundSurface

Table 605 – Metadata of ADEOfGroundSurface (DataType)

DEFINITION: ADEOfGroundSurface acts as a hook to define properties within an ADE that are to be added to a GroundSurface.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.15.5.13. ADEOfInteriorWallSurface

Table 606 – Metadata of ADEOfInteriorWallSurface (DataType)

DEFINITION: ADEOfInteriorWallSurface acts as a hook to define properties within an ADE that are to be added to an InteriorWallSurface.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.15.5.14. ADEOfOtherConstruction

Table 607 – Metadata of ADEOfOtherConstruction (DataType)

DEFINITION: ADEOfOtherConstruction acts as a hook to define properties within an ADE that are to be added to an OtherConstruction.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.15.5.15. ADEOfOuterCeilingSurface

Table 608 – Metadata of ADEOfOuterCeilingSurface (DataType)

DEFINITION:	ADEOfOuterCeilingSurface acts as a hook to define properties within an ADE that are to be added to an OuterCeilingSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.16. ADEOfOuterFloorSurface

Table 609 – Metadata of ADEOfOuterFloorSurface (DataType)

DEFINITION:	ADEOfOuterFloorSurface acts as a hook to define properties within an ADE that are to be added to an OuterFloorSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.17. ADEOfRoofSurface

Table 610 – Metadata of ADEOfRoofSurface (DataType)

DEFINITION:	ADEOfRoofSurface acts as a hook to define properties within an ADE that are to be added to a RoofSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.18. ADEOfWallSurface

Table 611 – Metadata of ADEOfWallSurface (DataType)

DEFINITION:	ADEOfWallSurface acts as a hook to define properties within an ADE that are to be added to a WallSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.19. ADEOfWindow

Table 612 – Metadata of ADEOfWindow (DataType)

DEFINITION:	ADEOfWindow acts as a hook to define properties within an ADE that are to be added to a Window.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.20. ADEOfWindowSurface

Table 613 – Metadata of ADEOfWindowSurface (DataType)

DEFINITION:	ADEOfWindowSurface acts as a hook to define properties within an ADE that are to be added to a WindowSurface.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.15.5.21. ConstructionEvent

Table 614 – Metadata of ConstructionEvent (DataType)

DEFINITION:	A ConstructionEvent is a data type used to describe a specific event that is associated with a construction. Examples are the issuing of a building permit or the renovation of a building.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 615 – Attributes of ConstructionEvent (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
dateOfEvent	Date [1..1]	Specifies the date at which the event took or will take place.
description	CharacterString [0..1]	Provides additional information on the event.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
event	EventValue [1..1]	Indicates the specific event type.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.5.22. Elevation

Table 616 – Metadata of Elevation (DataType)

DEFINITION:	Elevation is a data type that includes the elevation value itself and information on how this elevation was measured. [cf. INSPIRE]
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 617 – Attributes of Elevation (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
elevationReference	ElevationReference Value [1..1]	Specifies the level from which the elevation was measured. [cf. INSPIRE]
elevationValue	DirectPosition [1..1]	Specifies the value of the elevation. [cf. INSPIRE]

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.5.23. Height

Table 618 – Metadata of Height (DataType)

DEFINITION:	Height represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

Table 619 – Attributes of Height (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
highReference	ElevationReference Value [1..1]	Indicates the high point used to calculate the value of the height. [cf. INSPIRE]
lowReference	ElevationReference Value [1..1]	Indicates the low point used to calculate the value of the height. [cf. INSPIRE]
status	HeightStatusValue [1..1]	Indicates the way the height has been captured. [cf. INSPIRE]
value	Length [1..1]	Specifies the value of the height above or below ground. [cf. INSPIRE]

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.15.6. Enumerations

8.15.6.1. ConditionOfConstructionValue

Table 620 – Metadata of ConditionOfConstructionValue (Enumeration)

DEFINITION:	ConditionOfConstructionValue enumerates different conditions of a construction. [cf. INSPIRE]
STEREOTYPE:	«Enumeration»

Table 621 – Values of ConditionOfConstructionValue (Enumeration)

LITERAL VALUE	DEFINITION
declined	Indicates that the construction cannot be used under normal conditions, though its main elements (walls, roof) are still present. [cf. INSPIRE]
demolished	Indicates that the construction has been demolished. There are no more visible remains. [cf. INSPIRE]
functional	Indicates that the construction is functional. [cf. INSPIRE]
projected	Indicates that the construction is being designed. Construction works have not yet started. [cf. INSPIRE]
ruin	Indicates that the construction has been partly demolished and some main elements (roof, walls) have been destroyed. There are some visible remains of the construction. [cf. INSPIRE]
underConstruction	Indicates that the construction is under construction and not yet functional. This applies only to the initial construction works of the construction and not to maintenance work. [cf. INSPIRE]

8.15.6.2. HeightStatusValue

Table 622 – Metadata of HeightStatusValue (Enumeration)

DEFINITION:	HeightStatusValue enumerates the different methods used to capture a height. [cf. INSPIRE]
STEREOTYPE:	«Enumeration»

Table 623 – Values of HeightStatusValue (Enumeration)

LITERAL VALUE	DEFINITION
estimated	Indicates that the height has been estimated and not measured. [cf. INSPIRE]
measured	Indicates that the height has been (directly or indirectly) measured. [cf. INSPIRE]

8.15.6.3. RelationToConstruction

Table 624 – Metadata of RelationToConstruction (Enumeration)

DEFINITION:	RelationToConstruction is an enumeration used to describe whether an installation is positioned inside and/or outside of a construction.
STEREOTYPE:	«Enumeration»

Table 625 – Values of RelationToConstruction (Enumeration)

LITERAL VALUE	DEFINITION
inside	Indicates that the installation is positioned inside of the construction.
outside	Indicates that the installation is positioned outside of the construction.
bothInside AndOutside	Indicates that the installation is positioned inside as well as outside of the construction.

8.16. Bridge

Table 626 – Metadata of Bridge (ApplicationSchema)

DESCRIPTION:	The Bridge module supports representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.16.1. Classes

8.16.1.1. AbstractBridge

Table 627 – Metadata of AbstractBridge (FeatureType)

DEFINITION:	AbstractBridge is an abstract superclass representing the common attributes and associations of the classes Bridge and BridgePart.
SUBCLASS OF:	AbstractConstruction
STEREOTYPE:	«FeatureType»

Table 628 – Associations of AbstractBridge (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
bridgeRoom	BridgeRoom [1..*]	Relates the rooms to the Bridge or BridgePart.
bridgeFurniture	BridgeFurniture [1..*]	Relates the furniture objects to the Bridge or BridgePart.
bridge Constructive Element	BridgeConstructive Element [1..*]	Relates the constructive elements to the Bridge or BridgePart.
address	Address [1..*]	Relates the addresses to the Bridge or BridgePart.
bridge Installation	BridgeInstallation [1..*]	Relates the installation objects to the Bridge or BridgePart.

Table 629 – Attributes of AbstractBridge (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractBridge	ADEOfAbstractBridge [0..*]	Augments AbstractBridge with properties defined in an ADE.
class	BridgeClassValue [0..1]	Indicates the specific type of the Bridge or BridgePart.
function	BridgeFunctionValue [0..*]	Specifies the intended purposes of the Bridge or BridgePart.
isMovable	Boolean [0..1]	Indicates whether the Bridge or BridgePart can be moved to allow for watercraft to pass.
usage	BridgeUsageValue [0..*]	Specifies the actual uses of the Bridge or BridgePart.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.2. Bridge

Table 630 – Metadata of Bridge (TopLevelFeatureType)

DEFINITION:	A Bridge represents a structure that affords the passage of pedestrians, animals, vehicles, and service(s) above obstacles or between two points at a height above ground. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractBridge
STEREOTYPE:	«TopLevelFeatureType»

Table 631 – Associations of Bridge (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
bridgePart	BridgePart [1..*]	Relates the bridge parts to the Bridge.

Table 632 – Attributes of Bridge (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridge	ADEOfBridge [0..*]	Augments the Bridge with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.3. BridgeConstructiveElement

Table 633 – Metadata of BridgeConstructiveElement (FeatureType)

DEFINITION:	A BridgeConstructiveElement is an element of a bridge which is essential from a structural point of view. Examples are pylons, anchorages, slabs, beams.
SUBCLASS OF:	AbstractConstructiveElement
STEREOTYPE:	«FeatureType»

Table 634 – Attributes of BridgeConstructiveElement (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridge	ADEOfBridge	
Constructive Element	ConstructiveElement [0..*]	Augments the BridgeConstructiveElement with properties defined in an ADE.
class	BridgeConstructive ElementClassValue [0..1]	Indicates the specific type of the BridgeConstructiveElement.
function	BridgeConstructive ElementFunctionValue [0..*]	Specifies the intended purposes of the BridgeConstructiveElement.
usage	BridgeConstructive ElementUsageValue [0..*]	Specifies the actual uses of the BridgeConstructiveElement.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.4. BridgeFurniture

Table 635 – Metadata of BridgeFurniture (FeatureType)

DEFINITION:	A BridgeFurniture is an equipment for occupant use, usually not fixed to the bridge. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractFurniture
STEREOTYPE:	«FeatureType»

Table 636 – Attributes of BridgeFurniture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridgeFurniture	ADEOfBridgeFurniture [0..*]	Augments the BridgeFurniture with properties defined in an ADE.
class	BridgeFurnitureClass Value [0..1]	Indicates the specific type of the BridgeFurniture.
function	BridgeFurniture FunctionValue [0..*]	Specifies the intended purposes of the BridgeFurniture.
usage	BridgeFurnitureUsage Value [0..*]	Specifies the actual uses of the BridgeFurniture.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.5. BridgelInstallation

Table 637 – Metadata of BridgelInstallation (FeatureType)

DEFINITION:	A BridgelInstallation is a permanent part of a Bridge (inside and/or outside) which does not have the significance of a BridgePart. In contrast to BridgeConstructiveElements, a Bridge Installation is not essential from a structural point of view. Examples are stairs, antennas or railways.
SUBCLASS OF:	AbstractInstallation
STEREOTYPE:	«FeatureType»

Table 638 – Attributes of BridgelInstallation (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridgeInstallation	ADEOfBridgeInstallation [0..*]	Augments the BridgelInstallation with properties defined in an ADE.
class	BridgelInstallationClass Value [0..1]	Indicates the specific type of the BridgelInstallation.
function	BridgelInstallation FunctionValue [0..*]	Specifies the intended purposes of the BridgelInstallation.
usage	BridgelInstallation UsageValue [0..*]	Specifies the actual uses of the BridgelInstallation.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.6. BridgePart

Table 639 – Metadata of BridgePart (FeatureType)

DEFINITION:	A BridgePart is a physical or functional subdivision of a Bridge. It would be considered a Bridge, if it were not part of a collection of other BridgeParts.
SUBCLASS OF:	AbstractBridge
STEREOTYPE:	«FeatureType»

Table 640 – Attributes of BridgePart (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridgePart	ADEOfBridgePart [0..*]	Augments the BridgePart with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.1.7. BridgeRoom

Table 641 – Metadata of BridgeRoom (FeatureType)

DEFINITION:	A BridgeRoom is a space within a Bridge or BridgePart intended for human occupancy (e.g. a place of work or recreation) and/or containment (storage) of animals or things. A Bridge Room is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 642 – Associations of BridgeRoom (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the BridgeRoom. This relation is inherited from the Core module.
bridge Installation	BridgeInstallation [1..*]	Relates to the installation objects to the BridgeRoom.
bridgeFurniture	BridgeFurniture [1..*]	Relates the furniture objects to the BridgeRoom.

Table 643 – Attributes of BridgeRoom (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBridgeRoom	ADEOfBridgeRoom [0..*]	Augments the BridgeRoom with properties defined in an ADE.
class	BridgeRoomClassValue [0..1]	Indicates the specific type of the BridgeRoom.
function	BridgeRoomFunctionValue [0..*]	Specifies the intended purposes of the BridgeRoom.
usage	BridgeRoomUsageValue [0..*]	Specifies the actual uses of the BridgeRoom.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.16.2. Basic types

None.

8.16.3. Unions

None.

8.16.4. Code lists

8.16.4.1. BridgeClassValue

Table 644 – Metadata of BridgeClassValue (CodeList)

DEFINITION: BridgeClassValue is a code list used to further classify a Bridge.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.2. BridgeConstructiveElementClassValue

Table 645 – Metadata of BridgeConstructiveElementClassValue (CodeList)

DEFINITION:	BridgeConstructiveElementClassValue is a code list used to further classify a Bridge ConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.3. BridgeConstructiveElementFunctionValue

Table 646 – Metadata of BridgeConstructiveElementFunctionValue (CodeList)

DEFINITION:	BridgeConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BridgeConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.4. BridgeConstructiveElementUsageValue

Table 647 – Metadata of BridgeConstructiveElementUsageValue (CodeList)

DEFINITION:	BridgeConstructiveElementUsageValue is a code list that enumerates the different uses of a BridgeConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.5. BridgeFunctionValue

Table 648 – Metadata of BridgeFunctionValue (CodeList)

DEFINITION:	BridgeFunctionValue is a code list that enumerates the different purposes of a Bridge.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.6. BridgeFurnitureClassValue

Table 649 – Metadata of BridgeFurnitureClassValue (CodeList)

DEFINITION: BridgeFurnitureClassValue is a code list used to further classify a BridgeFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.7. BridgeFurnitureFunctionValue

Table 650 – Metadata of BridgeFurnitureFunctionValue (CodeList)

DEFINITION: BridgeFurnitureFunctionValue is a code list that enumerates the different purposes of a BridgeFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.8. BridgeFurnitureUsageValue

Table 651 – Metadata of BridgeFurnitureUsageValue (CodeList)

DEFINITION: BridgeFurnitureUsageValue is a code list that enumerates the different uses of a BridgeFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.9. BridgeInstallationClassValue

Table 652 – Metadata of BridgeInstallationClassValue (CodeList)

DEFINITION: BridgeInstallationClassValue is a code list used to further classify a BridgeInstallation.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.10. BridgeInstallationFunctionValue

Table 653 – Metadata of BridgeInstallationFunctionValue (CodeList)

DEFINITION: BridgeInstallationFunctionValue is a code list that enumerates the different purposes of a BridgeInstallation.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.11. BridgeInstallationUsageValue

Table 654 – Metadata of BridgeInstallationUsageValue (CodeList)

DEFINITION: BridgeInstallationUsageValue is a code list that enumerates the different uses of a Bridge Installation.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.12. BridgeRoomClassValue

Table 655 – Metadata of BridgeRoomClassValue (CodeList)

DEFINITION: BridgeRoomClassValue is a code list used to further classify a BridgeRoom.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.16.4.13. BridgeRoomFunctionValue

Table 656 – Metadata of BridgeRoomFunctionValue (CodeList)

DEFINITION:	BridgeRoomFunctionValue is a code list that enumerates the different purposes of a Bridge Room.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.14. BridgeRoomUsageValue

Table 657 – Metadata of BridgeRoomUsageValue (CodeList)

DEFINITION:	BridgeRoomUsageValue is a code list that enumerates the different uses of a BridgeRoom.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.4.15. BridgeUsageValue

Table 658 – Metadata of BridgeUsageValue (CodeList)

DEFINITION:	BridgeUsageValue is a code list that enumerates the different uses of a Bridge.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.16.5. Data types

8.16.5.1. ADEOfAbstractBridge

Table 659 – Metadata of ADEOfAbstractBridge (DataType)

DEFINITION:	ADEOfAbstractBridge acts as a hook to define properties within an ADE that are to be added to AbstractBridge.
SUBCLASS OF:	None

STEREOTYPE: «DataType»

8.16.5.2. ADEOfBridge

Table 660 – Metadata of ADEOfBridge (DataType)

DEFINITION: ADEOfBridge acts as a hook to define properties within an ADE that are to be added to a Bridge.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.16.5.3. ADEOfBridgeConstructiveElement

Table 661 – Metadata of ADEOfBridgeConstructiveElement (DataType)

DEFINITION: ADEOfBridgeConstructiveElement acts as a hook to define properties within an ADE that are to be added to a BridgeConstructiveElement.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.16.5.4. ADEOfBridgeFurniture

Table 662 – Metadata of ADEOfBridgeFurniture (DataType)

DEFINITION: ADEOfBridgeFurniture acts as a hook to define properties within an ADE that are to be added to a BridgeFurniture.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.16.5.5. ADEOfBridgeInstallation

Table 663 – Metadata of ADEOfBridgeInstallation (DataType)

DEFINITION:	ADEOfBridgeInstallation acts as a hook to define properties within an ADE that are to be added to a BridgeInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.16.5.6. ADEOfBridgePart

Table 664 – Metadata of ADEOfBridgePart (DataType)

DEFINITION:	ADEOfBridgePart acts as a hook to define properties within an ADE that are to be added to a BridgePart.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.16.5.7. ADEOfBridgeRoom

Table 665 – Metadata of ADEOfBridgeRoom (DataType)

DEFINITION:	ADEOfBridgeRoom acts as a hook to define properties within an ADE that are to be added to a BridgeRoom.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.16.6. Enumerations

None.

8.17. Building

Table 666 – Metadata of Building (ApplicationSchema)

DESCRIPTION:	The Building module supports representation of thematic and spatial aspects of buildings, building parts, building installations, building subdivisions, and interior building structures.
PARENT PACKAGE:	CityGML
STEREOTYPE:	«ApplicationSchema»

8.17.1. Classes

8.17.1.1. AbstractBuilding

Table 667 – Metadata of AbstractBuilding (FeatureType)

DEFINITION:	AbstractBuilding is an abstract superclass representing the common attributes and associations of the classes Building and BuildingPart.
SUBCLASS OF:	AbstractConstruction
STEREOTYPE:	«FeatureType»

Table 668 – Associations of AbstractBuilding (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
building Installation	BuildingInstallation [1..*]	Relates the installation objects to the Building or BuildingPart.
buildingFurniture	BuildingFurniture [1..*]	Relates the furniture objects to the Building or BuildingPart.
building Constructive Element	BuildingConstructive Element [1..*]	Relates the constructive elements to the Building or BuildingPart.
address	Address [1..*]	Relates the addresses to the Building or BuildingPart.
buildingRoom	BuildingRoom [1..*]	Relates the rooms to the Building or BuildingPart.
building Subdivision	AbstractBuilding Subdivision [1..*]	Relates the logical subdivisions to the Building or BuildingPart.

Table 669 – Attributes of AbstractBuilding (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstractBuilding	ADEOfAbstractBuilding [0..*]	Augments AbstractBuilding with properties defined in an ADE.
class	BuildingClassValue [0..1]	Indicates the specific type of the Building or BuildingPart.
function	BuildingFunctionValue [0..*]	Specifies the intended purposes of the Building or BuildingPart.
roofType	RoofTypeValue [0..1]	Indicates the shape of the roof of the Building or BuildingPart.
storeyHeightsAboveGround	MeasureOrNilReasonList [0..1]	Lists the heights of each storey above ground. The first value in the list denotes the height of the storey closest to the ground level, the last value denotes the height furthest away.
storeyHeightsBelowGround	MeasureOrNilReasonList [0..1]	Lists the height of each storey below ground. The first value in the list denotes the height of the storey closest to the ground level, the last value denotes the height furthest away.
storeysAboveGround	Integer [0..1]	Indicates the number of storeys positioned above ground level.
storeysBelowGround	Integer [0..1]	Indicates the number of storeys positioned below ground level.
usage	BuildingUsageValue [0..*]	Specifies the actual uses of the Building or BuildingPart.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.2. AbstractBuildingSubdivision

Table 670 – Metadata of AbstractBuildingSubdivision (FeatureType)

DEFINITION:	AbstractBuildingSubdivision is the abstract superclass for different kinds of logical building subdivisions.
SUBCLASS OF:	AbstractLogicalSpace
STEREOTYPE:	«FeatureType»

Table 671 – Associations of AbstractBuildingSubdivision (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
buildingFurniture	BuildingFurniture [1..*]	Relates the furniture objects to the building subdivision.

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
building Constructive Element	BuildingConstructive Element [1..*]	Relates the constructive elements to the building subdivision.
building Installation	BuildingInstallation [1..*]	Relates the installation objects to the building subdivision.
buildingRoom	BuildingRoom [1..*]	Relates the rooms to the building subdivision.

Table 672 – Attributes of AbstractBuildingSubdivision (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Building Subdivision	ADEOfAbstract BuildingSubdivision [0..*]	Augments AbstractBuildingSubdivision with properties defined in an ADE.
class	BuildingSubdivision ClassValue [0..1]	Indicates the specific type of the building subdivision.
elevation	Elevation [0..*]	Specifies qualified elevations of the building subdivision in relation to a well-defined surface which is commonly taken as origin (e.g. geoid or water level). [cf. INSPIRE]
function	BuildingSubdivision FunctionValue [0..*]	Specifies the intended purposes of the building subdivision.
sortKey	Real [0..1]	Defines an order among the objects that belong to the building subdivision. An example is the sorting of storeys.
usage	BuildingSubdivision UsageValue [0..*]	Specifies the actual uses of the building subdivision.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.3. Building

Table 673 – Metadata of Building (TopLevelFeatureType)

DEFINITION:	A Building is a free-standing, self-supporting construction that is roofed, usually walled, and can be entered by humans and is normally designed to stand permanently in one place. It is intended for human occupancy (e.g. a place of work or recreation), habitation and/or shelter of humans, animals or things.
SUBCLASS OF:	AbstractBuilding
STEREOTYPE:	«TopLevelFeatureType»

Table 674 – Associations of Building (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
buildingPart	BuildingPart [1..*]	Relates the building parts to the Building.

Table 675 – Attributes of Building (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuilding	ADEOfBuilding [0..*]	Augments the Building with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.4. BuildingConstructiveElement

Table 676 – Metadata of BuildingConstructiveElement (FeatureType)

DEFINITION:	A BuildingConstructiveElement is an element of a Building which is essential from a structural point of view. Examples are walls, slabs, staircases, beams.
SUBCLASS OF:	AbstractConstructiveElement
STEREOTYPE:	«FeatureType»

Table 677 – Attributes of BuildingConstructiveElement (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuilding	ADEOfBuilding	
Constructive Element	ConstructiveElement [0..*]	Augments the BuildingConstructiveElement with properties defined in an ADE.
class	BuildingConstructiveElementClassValue [0..1]	Indicates the specific type of the BuildingConstructiveElement.
function	BuildingConstructiveElementFunctionValue [0..*]	Specifies the intended purposes of the BuildingConstructiveElement.
usage	BuildingConstructiveElementUsageValue [0..*]	Specifies the actual uses of the BuildingConstructiveElement.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.5. BuildingFurniture

Table 678 – Metadata of BuildingFurniture (FeatureType)

DEFINITION:	A BuildingFurniture is an equipment for occupant use, usually not fixed to the building. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractFurniture
STEREOTYPE:	«FeatureType»

Table 679 – Attributes of BuildingFurniture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuildingFurniture	ADEOfBuildingFurniture [0..*]	Augments the BuildingFurniture with properties defined in an ADE.
class	BuildingFurnitureClass Value [0..1]	Indicates the specific type of the BuildingFurniture.
function	BuildingFurniture FunctionValue [0..*]	Specifies the intended purposes of the BuildingFurniture.
usage	BuildingFurniture UsageValue [0..*]	Specifies the actual uses of the BuildingFurniture.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.6. BuildingInstallation

Table 680 – Metadata of BuildingInstallation (FeatureType)

DEFINITION:	A BuildingInstallation is a permanent part of a Building (inside and/or outside) which has not the significance of a BuildingPart. Examples are stairs, antennas, balconies or small roofs.
SUBCLASS OF:	AbstractInstallation
STEREOTYPE:	«FeatureType»

Table 681 – Attributes of BuildingInstallation (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuildingInstallation	ADEOfBuildingInstallation [0..*]	Augments the BuildingInstallation with properties defined in an ADE.
class	BuildingInstallationClassValue [0..1]	Indicates the specific type of the BuildingInstallation.
function	BuildingInstallationFunctionValue [0..*]	Specifies the intended purposes of the BuildingInstallation.
usage	BuildingInstallationUsageValue [0..*]	Specifies the actual uses of the BuildingInstallation.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.7. BuildingPart

Table 682 – Metadata of BuildingPart (FeatureType)

DEFINITION:	A BuildingPart is a physical or functional subdivision of a Building. It would be considered a Building, if it were not part of a collection of other BuildingParts.
SUBCLASS OF:	AbstractBuilding
STEREOTYPE:	«FeatureType»

Table 683 – Attributes of BuildingPart (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuildingPart	ADEOfBuildingPart [0..*]	Augments the BuildingPart with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.8. BuildingRoom

Table 684 – Metadata of BuildingRoom (FeatureType)

DEFINITION:	A BuildingRoom is a space within a Building or BuildingPart intended for human occupancy (e.g. a place of work or recreation) and/or containment of animals or things. A BuildingRoom is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
-------------	--

SUBCLASS OF: AbstractUnoccupiedSpace

STEREOTYPE: «FeatureType»

Table 685 – Associations of BuildingRoom (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the BuildingRoom. This relation is inherited from the Core module.
buildingFurniture	BuildingFurniture [1..*]	Relates the furniture objects to the BuildingRoom.
building Installation	BuildingInstallation [1..*]	Relates the installation objects to the BuildingRoom.

Table 686 – Attributes of BuildingRoom (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuildingRoom	ADEOfBuildingRoom [0..*]	Augments the BuildingRoom with properties defined in an ADE.
class	BuildingRoomClass Value [0..1]	Indicates the specific type of the BuildingRoom.
function	BuildingRoomFunction Value [0..*]	Specifies the intended purposes of the BuildingRoom.
roomHeight	RoomHeight [0..*]	Specifies qualified heights of the BuildingRoom.
usage	BuildingRoomUsage Value [0..*]	Specifies the actual uses of the BuildingRoom.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.9. BuildingUnit

Table 687 – Metadata of BuildingUnit (FeatureType)

DEFINITION: A BuildingUnit is a logical subdivision of a Building. BuildingUnits are formed according to some homogeneous property like function, ownership, management, or accessibility. They may be separately sold, rented out, inherited, managed, etc.

SUBCLASS OF: AbstractBuildingSubdivision

STEREOTYPE: «FeatureType»

Table 688 – Associations of BuildingUnit (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
address	Address [1..*]	Relates to the addresses that are assigned to the BuildingUnit.
storey	Storey [1..*]	Relates to the storeys on which the BuildingUnit is located.

Table 689 – Attributes of BuildingUnit (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfBuildingUnit	ADEOfBuildingUnit [0..*]	Augments the BuildingUnit with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.1.10. Storey

Table 690 – Metadata of Storey (FeatureType)

DEFINITION:	A Storey is typically a horizontal section of a Building. Storeys are not always defined according to the building structure, but can also be defined according to logical considerations.
SUBCLASS OF:	AbstractBuildingSubdivision
STEREOTYPE:	«FeatureType»

Table 691 – Associations of Storey (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
boundary	AbstractThematicSurface [1..*]	Relates to the surfaces that bound the Storey. This relation is inherited from the Core module.
buildingUnit	BuildingUnit [1..*]	Relates to the building units that belong to the Storey.

Table 692 – Attributes of Storey (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfStorey	ADEOfStorey [0..*]	Augments the Storey with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.2. Basic types

None.

8.17.3. Unions

None.

8.17.4. Code lists

8.17.4.1. BuildingClassValue

Table 693 – Metadata of BuildingClassValue (CodeList)

DEFINITION: BuildingClassValue is a code list used to further classify a Building.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.2. BuildingConstructiveElementClassValue

Table 694 – Metadata of BuildingConstructiveElementClassValue (CodeList)

DEFINITION: BuildingConstructiveElementClassValue is a code list used to further classify a Building ConstructiveElement.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.3. BuildingConstructiveElementFunctionValue

Table 695 – Metadata of BuildingConstructiveElementFunctionValue (CodeList)

DEFINITION:	BuildingConstructiveElementFunctionValue is a code list that enumerates the different purposes of a BuildingConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.4. BuildingConstructiveElementUsageValue

Table 696 – Metadata of BuildingConstructiveElementUsageValue (CodeList)

DEFINITION:	BuildingConstructiveElementUsageValue is a code list that enumerates the different uses of a BuildingConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.5. BuildingFunctionValue

Table 697 – Metadata of BuildingFunctionValue (CodeList)

DEFINITION:	BuildingFunctionValue is a code list that enumerates the different purposes of a Building.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.6. BuildingFurnitureClassValue

Table 698 – Metadata of BuildingFurnitureClassValue (CodeList)

DEFINITION:	BuildingFurnitureClassValue is a code list used to further classify a BuildingFurniture.
-------------	--

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.7. BuildingFurnitureFunctionValue

Table 699 – Metadata of BuildingFurnitureFunctionValue (CodeList)

DEFINITION: BuildingFurnitureFunctionValue is a code list that enumerates the different purposes of a BuildingFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.8. BuildingFurnitureUsageValue

Table 700 – Metadata of BuildingFurnitureUsageValue (CodeList)

DEFINITION: BuildingFurnitureUsageValue is a code list that enumerates the different uses of a Building Furniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.9. BuildingInstallationClassValue

Table 701 – Metadata of BuildingInstallationClassValue (CodeList)

DEFINITION: BuildingInstallationClassValue is a code list used to further classify a BuildingInstallation.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.10. BuildingInstallationFunctionValue

Table 702 – Metadata of BuildingInstallationFunctionValue (CodeList)

DEFINITION:	BuildingInstallationFunctionValue is a code list that enumerates the different purposes of a BuildingInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.11. BuildingInstallationUsageValue

Table 703 – Metadata of BuildingInstallationUsageValue (CodeList)

DEFINITION:	BuildingInstallationUsageValue is a code list that enumerates the different uses of a BuildingInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.12. BuildingRoomClassValue

Table 704 – Metadata of BuildingRoomClassValue (CodeList)

DEFINITION:	BuildingRoomClassValue is a code list used to further classify a BuildingRoom.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.13. BuildingRoomFunctionValue

Table 705 – Metadata of BuildingRoomFunctionValue (CodeList)

DEFINITION:	BuildingRoomFunctionValue is a code list that enumerates the different purposes of a BuildingRoom.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.14. BuildingRoomUsageValue

Table 706 – Metadata of BuildingRoomUsageValue (CodeList)

DEFINITION:	BuildingRoomUsageValue is a code list that enumerates the different uses of a Building Room.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.15. BuildingSubdivisionClassValue

Table 707 – Metadata of BuildingSubdivisionClassValue (CodeList)

DEFINITION:	BuildingSubdivisionClassValue is a code list used to further classify a BuildingSubdivision.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.16. BuildingSubdivisionFunctionValue

Table 708 – Metadata of BuildingSubdivisionFunctionValue (CodeList)

DEFINITION:	BuildingSubdivisionFunctionValue is a code list that enumerates the different purposes of a BuildingSubdivision.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.17.4.17. BuildingSubdivisionUsageValue

Table 709 – Metadata of BuildingSubdivisionUsageValue (CodeList)

DEFINITION:	BuildingSubdivisionUsageValue is a code list that enumerates the different uses of a Building Subdivision.
--------------------	--

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.18. BuildingUsageValue

Table 710 – Metadata of BuildingUsageValue (CodeList)

DEFINITION: BuildingUsageValue is a code list that enumerates the different uses of a Building.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.19. RoofTypeValue

Table 711 – Metadata of RoofTypeValue (CodeList)

DEFINITION: RoofTypeValue is a code list that enumerates different roof types.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.4.20. RoomElevationReferenceValue

Table 712 – Metadata of RoomElevationReferenceValue (CodeList)

DEFINITION: RoomElevationReferenceValue is a code list that enumerates the different elevation reference levels used to measure room heights.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.17.5. Data types

8.17.5.1. ADEOfAbstractBuilding

Table 713 – Metadata of ADEOfAbstractBuilding (DataType)

DEFINITION:	ADEOfAbstractBuilding acts as a hook to define properties within an ADE that are to be added to AbstractBuilding.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.2. ADEOfAbstractBuildingSubdivision

Table 714 – Metadata of ADEOfAbstractBuildingSubdivision (DataType)

DEFINITION:	ADEOfAbstractBuildingSubdivision acts as a hook to define properties within an ADE that are to be added to AbstractBuildingSubdivision.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.3. ADEOfBuilding

Table 715 – Metadata of ADEOfBuilding (DataType)

DEFINITION:	ADEOfBuilding acts as a hook to define properties within an ADE that are to be added to a Building.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.4. ADEOfBuildingConstructiveElement

Table 716 – Metadata of ADEOfBuildingConstructiveElement (DataType)

DEFINITION:	ADEOfBuildingConstructiveElement acts as a hook to define properties within an ADE that are to be added to a BuildingConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.5. ADEOfBuildingFurniture

Table 717 – Metadata of ADEOfBuildingFurniture (DataType)

DEFINITION:	ADEOfBuildingFurniture acts as a hook to define properties within an ADE that are to be added to a BuildingFurniture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.6. ADEOfBuildingInstallation

Table 718 – Metadata of ADEOfBuildingInstallation (DataType)

DEFINITION:	ADEOfBuildingInstallation acts as a hook to define properties within an ADE that are to be added to a BuildingInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.7. ADEOfBuildingPart

Table 719 – Metadata of ADEOfBuildingPart (DataType)

DEFINITION:	ADEOfBuildingPart acts as a hook to define properties within an ADE that are to be added to a BuildingPart.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.8. ADEOfBuildingRoom

Table 720 – Metadata of ADEOfBuildingRoom (DataType)

DEFINITION:	ADEOfBuildingRoom acts as a hook to define properties within an ADE that are to be added to a BuildingRoom.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.9. ADEOfBuildingUnit

Table 721 – Metadata of ADEOfBuildingUnit (DataType)

DEFINITION:	ADEOfBuildingUnit acts as a hook to define properties within an ADE that are to be added to a BuildingUnit.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.10. ADEOfStorey

Table 722 – Metadata of ADEOfStorey (DataType)

DEFINITION:	ADEOfStorey acts as a hook to define properties within an ADE that are to be added to a Storey.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.17.5.11. RoomHeight

Table 723 – Metadata of RoomHeight (DataType)

DEFINITION:	The RoomHeight represents a vertical distance (measured or estimated) between a low reference and a high reference. [cf. INSPIRE]
--------------------	---

SUBCLASS OF: None

STEREOTYPE: «DataType»

Table 724 – Attributes of RoomHeight (DataType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
highReference	RoomElevation ReferenceValue [1..1]	Indicates the high point used to calculate the value of the room height.
lowReference	RoomElevation ReferenceValue [1..1]	Indicates the low point used to calculate the value of the room height.
status	HeightStatusValue [1..1]	Indicates the way the room height has been captured.
value	Length [1..1]	Specifies the value of the room height.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.17.6. Enumerations

None.

8.18. Tunnel

Table 725 – Metadata of Tunnel (ApplicationSchema)

DESCRIPTION: The Tunnel module supports representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures.

PARENT PACKAGE: CityGML

STEREOTYPE: «ApplicationSchema»

8.18.1. Classes

8.18.1.1. AbstractTunnel

Table 726 – Metadata of AbstractTunnel (FeatureType)

DEFINITION:	AbstractTunnel is an abstract superclass representing the common attributes and associations of the classes Tunnel and TunnelPart.
SUBCLASS OF:	AbstractConstruction
STEREOTYPE:	«FeatureType»

Table 727 – Associations of AbstractTunnel (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
tunnel Constructive Element	TunnelConstructive Element [1..*]	Relates the constructive elements to the Tunnel or TunnelPart.
tunnelFurniture	TunnelFurniture [1..*]	Relates the furniture objects to the Tunnel or TunnelPart.
tunnel Installation	TunnelInstallation [1..*]	Relates the installation objects to the Tunnel or TunnelPart.
hollowSpace	HollowSpace [1..*]	Relates the hollow spaces to the Tunnel or TunnelPart.

Table 728 – Attributes of AbstractTunnel (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfAbstract Tunnel	ADEOfAbstractTunnel [0..*]	Augments AbstractTunnel with properties defined in an ADE.
class	TunnelClassValue [0..1]	Indicates the specific type of the Tunnel or TunnelPart.
function	TunnelFunctionValue [0..*]	Specifies the intended purposes of the Tunnel or TunnelPart.
usage	TunnelUsageValue [0.. *]	Specifies the actual uses of the Tunnel or TunnelPart.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.2. HollowSpace

Table 729 – Metadata of HollowSpace (FeatureType)

DEFINITION:	A HollowSpace is a space within a Tunnel or TunnelPart intended for certain functions (e.g. transport or passage ways, service rooms, emergency shelters). A HollowSpace is bounded physically and/or virtually (e.g. by ClosureSurfaces or GenericSurfaces).
SUBCLASS OF:	AbstractUnoccupiedSpace
STEREOTYPE:	«FeatureType»

Table 730 – Associations of HollowSpace (FeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
tunnel Installation	TunnelInstallation [1..*]	Relates the installation objects to the HollowSpace.
tunnelFurniture	TunnelFurniture [1..*]	Relates the furniture objects to the HollowSpace.
boundary	AbstractThematic Surface [1..*]	Relates to the surfaces that bound the HollowSpace. This relation is inherited from the Core module.

Table 731 – Attributes of HollowSpace (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfHollow Space	ADEOfHollowSpace [0..*]	Augments the HollowSpace with properties defined in an ADE.
class	HollowSpaceClass Value [0..1]	Indicates the specific type of the HollowSpace.
function	HollowSpaceFunction Value [0..*]	Specifies the intended purposes of the HollowSpace.
usage	HollowSpaceUsage Value [0..*]	Specifies the actual uses of the HollowSpace.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.3. Tunnel

Table 732 – Metadata of Tunnel (TopLevelFeatureType)

DEFINITION:	A Tunnel represents a horizontal or sloping enclosed passage way of a certain length, mainly underground or underwater. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractTunnel

STEREOTYPE: «TopLevelFeatureType»

Table 733 – Associations of Tunnel (TopLevelFeatureType)

ROLE NAME	TARGET CLASS AND MULTIPLICITY	DEFINITION
tunnelPart	TunnelPart [1..*]	Relates the tunnel parts to the Tunnel.

Table 734 – Attributes of Tunnel (TopLevelFeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTunnel	ADEOfTunnel [0..*]	Augments the Tunnel with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.4. TunnelConstructiveElement

Table 735 – Metadata of TunnelConstructiveElement (FeatureType)

DEFINITION:	A TunnelConstructiveElement is an element of a Tunnel which is essential from a structural point of view. Examples are walls, slabs, beams.
SUBCLASS OF:	AbstractConstructiveElement
STEREOTYPE:	«FeatureType»

Table 736 – Attributes of TunnelConstructiveElement (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTunnel	ADEOfTunnel	
Constructive Element	ConstructiveElement [0..*]	Augments the TunnelConstructiveElement with properties defined in an ADE.
class	TunnelConstructiveElementClassValue [0..1]	Indicates the specific type of the TunnelConstructiveElement.
function	TunnelConstructiveElementFunctionValue [0..*]	Specifies the intended purposes of the TunnelConstructiveElement.

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
usage	TunnelConstructiveElementUsageValue [0..*]	Specifies the actual uses of the TunnelConstructiveElement.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.5. TunnelFurniture

Table 737 – Metadata of TunnelFurniture (FeatureType)

DEFINITION:	A TunnelFurniture is an equipment for occupant use, usually not fixed to the tunnel. [cf. ISO 6707-1]
SUBCLASS OF:	AbstractFurniture
STEREOTYPE:	«FeatureType»

Table 738 – Attributes of TunnelFurniture (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTunnelFurniture	ADEOfTunnelFurnitureValue [0..*]	Augments the TunnelFurniture with properties defined in an ADE.
class	TunnelFurnitureClassValue [0..1]	Indicates the specific type of the TunnelFurniture.
function	TunnelFurnitureFunctionValue [0..*]	Specifies the intended purposes of the TunnelFurniture.
usage	TunnelFurnitureUsageValue [0..*]	Specifies the actual uses of the TunnelFurniture.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.6. TunnellInstallation

Table 739 – Metadata of TunnellInstallation (FeatureType)

DEFINITION:	A TunnellInstallation is a permanent part of a Tunnel (inside and/or outside) which does not have the significance of a TunnelPart. In contrast to TunnelConstructiveElement, a Tunnel Installation is not essential from a structural point of view. Examples are stairs, antennas or railings.
-------------	--

SUBCLASS OF: AbstractInstallation

STEREOTYPE: «FeatureType»

Table 740 – Attributes of TunnellInstallation (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTunnelInstallation	ADEOfTunnelInstallation [0..*]	Augments the TunnellInstallation with properties defined in an ADE.
class	TunnellInstallationClass Value [0..1]	Indicates the specific type of the TunnellInstallation.
function	TunnellInstallation FunctionValue [0..*]	Specifies the intended purposes of the TunnellInstallation.
usage	TunnellInstallation UsageValue [0..*]	Specifies the actual uses of the TunnellInstallation.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.1.7. TunnelPart

Table 741 – Metadata of TunnelPart (FeatureType)

DEFINITION: A TunnelPart is a physical or functional subdivision of a Tunnel. It would be considered a Tunnel, if it were not part of a collection of other TunnelParts.

SUBCLASS OF: AbstractTunnel

STEREOTYPE: «FeatureType»

Table 742 – Attributes of TunnelPart (FeatureType)

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
adeOfTunnelPart	ADEOfTunnelPart [0..*]	Augments the TunnelPart with properties defined in an ADE.

NOTE: Unless otherwise specified, all attributes and role names have the stereotype «Property».

8.18.2. Basic types

None.

8.18.3. Unions

None.

8.18.4. Code lists

8.18.4.1. HollowSpaceClassValue

Table 743 – Metadata of HollowSpaceClassValue (CodeList)

DEFINITION: HollowSpaceClassValue is a code list used to further classify a HollowSpace.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.2. HollowSpaceFunctionValue

Table 744 – Metadata of HollowSpaceFunctionValue (CodeList)

DEFINITION: HollowSpaceFunctionValue is a code list that enumerates the different purposes of a HollowSpace.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.3. HollowSpaceUsageValue

Table 745 – Metadata of HollowSpaceUsageValue (CodeList)

DEFINITION: HollowSpaceUsageValue is a code list that enumerates the different uses of a HollowSpace.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.4. TunnelClassValue

Table 746 – Metadata of TunnelClassValue (CodeList)

DEFINITION: TunnelClassValue is a code list used to further classify a Tunnel.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.5. TunnelConstructiveElementClassValue

Table 747 – Metadata of TunnelConstructiveElementClassValue (CodeList)

DEFINITION: TunnelConstructiveElementClassValue is a code list used to further classify a Tunnel ConstructiveElement.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.6. TunnelConstructiveElementFunctionValue

Table 748 – Metadata of TunnelConstructiveElementFunctionValue (CodeList)

DEFINITION: TunnelConstructiveElementFunctionValue is a code list that enumerates the different purposes of a TunnelConstructiveElement.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.7. TunnelConstructiveElementUsageValue

Table 749 – Metadata of TunnelConstructiveElementUsageValue (CodeList)

DEFINITION: TunnelConstructiveElementUsageValue is a code list that enumerates the different uses of a TunnelConstructiveElement.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.8. TunnelFunctionValue

Table 750 – Metadata of TunnelFunctionValue (CodeList)

DEFINITION: TunnelFunctionValue is a code list that enumerates the different purposes of a Tunnel.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.9. TunnelFurnitureClassValue

Table 751 – Metadata of TunnelFurnitureClassValue (CodeList)

DEFINITION: TunnelFurnitureClassValue is a code list used to further classify a TunnelFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.10. TunnelFurnitureFunctionValue

Table 752 – Metadata of TunnelFurnitureFunctionValue (CodeList)

DEFINITION: TunnelFurnitureFunctionValue is a code list that enumerates the different purposes of a TunnelFurniture.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.4.11. TunnelFurnitureUsageValue

Table 753 – Metadata of TunnelFurnitureUsageValue (CodeList)

DEFINITION:	TunnelFurnitureUsageValue is a code list that enumerates the different uses of a Tunnel Furniture.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.18.4.12. TunnelInstallationClassValue

Table 754 – Metadata of TunnelInstallationClassValue (CodeList)

DEFINITION:	TunnelInstallationClassValue is a code list used to further classify a TunnelInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.18.4.13. TunnelInstallationFunctionValue

Table 755 – Metadata of TunnelInstallationFunctionValue (CodeList)

DEFINITION:	TunnelInstallationFunctionValue is a code list that enumerates the different purposes of a TunnelInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.18.4.14. TunnelInstallationUsageValue

Table 756 – Metadata of TunnelInstallationUsageValue (CodeList)

DEFINITION:	TunnelInstallationUsageValue is a code list that enumerates the different uses of a Tunnel Installation.
SUBCLASS OF:	None
STEREOTYPE:	«CodeList»

8.18.4.15. TunnelUsageValue

Table 757 – Metadata of TunnelUsageValue (CodeList)

DEFINITION: TunnelUsageValue is a code list that enumerates the different uses of a Tunnel.

SUBCLASS OF: None

STEREOTYPE: «CodeList»

8.18.5. Data types

8.18.5.1. ADEOfAbstractTunnel

Table 758 – Metadata of ADEOfAbstractTunnel (DataType)

DEFINITION: ADEOfAbstractTunnel acts as a hook to define properties within an ADE that are to be added to AbstractTunnel.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.18.5.2. ADEOfHollowSpace

Table 759 – Metadata of ADEOfHollowSpace (DataType)

DEFINITION: ADEOfHollowSpace acts as a hook to define properties within an ADE that are to be added to a HollowSpace.

SUBCLASS OF: None

STEREOTYPE: «DataType»

8.18.5.3. ADEOfTunnel

Table 760 – Metadata of ADEOfTunnel (DataType)

DEFINITION:	ADEOfTunnel acts as a hook to define properties within an ADE that are to be added to a Tunnel.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.18.5.4. ADEOfTunnelConstructiveElement

Table 761 – Metadata of ADEOfTunnelConstructiveElement (DataType)

DEFINITION:	ADEOfTunnelConstructiveElement acts as a hook to define properties within an ADE that are to be added to a TunnelConstructiveElement.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.18.5.5. ADEOfTunnelFurniture

Table 762 – Metadata of ADEOfTunnelFurniture (DataType)

DEFINITION:	ADEOfTunnelFurniture acts as a hook to define properties within an ADE that are to be added to a TunnelFurniture.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.18.5.6. ADEOfTunnellInstallation

Table 763 – Metadata of ADEOfTunnellInstallation (DataType)

DEFINITION:	ADEOfTunnellInstallation acts as a hook to define properties within an ADE that are to be added to a TunnellInstallation.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.18.5.7. ADEOfTunnelPart

Table 764 – Metadata of ADEOfTunnelPart (DataType)

DEFINITION:	ADEOfTunnelPart acts as a hook to define properties within an ADE that are to be added to a TunnelPart.
SUBCLASS OF:	None
STEREOTYPE:	«DataType»

8.18.6. Enumerations

None.

9

APPLICATION DOMAIN EXTENSION (ADE)

APPLICATION DOMAIN EXTENSION (ADE)

An *Application Domain Extension* (ADE) is a formal and systematic extension of the CityGML Conceptual Model (CM) for a specific application or domain. The ADE is expressed in the form of a UML conceptual model. The domain data is mapped to a set of additional classes, attributes, and relations. ADEs may use elements from the CityGML CM to derive application-specific subclasses, to inject additional properties, to associate application data with predefined CityGML content, or to define value domains for attributes.

The ADE mechanism allows application-specific information to be aligned with the CityGML CM in a well-structured and systematic way. By this means, CityGML can be extended to meet the information needs of an application while at the same time preserving its concepts and semantic structures. Moreover, and in contrast to generic city objects and attributes, application data can be validated against the formal definition of an ADE to ensure semantic interoperability.

Previous versions of the CityGML Standard defined the ADE mechanism solely at the level of the XML Schema encoding. With CityGML 3.0, ADEs become platform-independent models at a conceptual level that can be mapped to multiple and different target encodings.

ADEs have successfully been implemented in practice and enable a wide range of applications and use cases based on the CityGML Standard. An overview and discussion of existing ADEs is provided in [Biljecki et al. 2018].

9.1. General Rules for ADEs

An ADE shall be defined as a UML conceptual model in accordance with the General Feature Model and the rules for creating application schemas in UML as specified in ISO 19109:2015 and the rules and constraints for using UML to model geographic information as specified in ISO 19103:2015. The UML notations and stereotypes (Clause 5.2) used in the CityGML conceptual model should also be applied to corresponding model elements in an ADE.

Every ADE shall be organized into one or more UML packages having globally unique namespaces and containing all UML model elements defined by the ADE. An ADE may additionally import and use predefined classes from external conceptual UML models such as the CityGML modules or the standardized schemas of the ISO 19100 series of International Standards.

9.2. Defining New ADE Model Elements

Following ISO 19109, the primary view of geospatial information and the core element of application schemas is the *feature*. ADEs therefore typically extend CityGML by defining new

feature types appropriate to the application area together with additional content such as object types, data types, code lists, and enumerations.

Every feature type in an ADE shall be derived either directly or indirectly from the CityGML root feature type *Core::AbstractFeature* or, depending on its type and characteristics, from a more appropriate subclass thereof. According to the general CityGML space concept, features representing spaces or space boundaries shall be derived either directly or indirectly from *Core::AbstractSpace* or *Core::AbstractSpaceBoundary* respectively. UML classes representing top-level feature types shall use the «*TopLevelFeatureType*» stereotype.

In contrast to feature types, object types and data types are not required to be derived from a predefined CityGML class unless explicitly stated otherwise.

ADE classes may have an unlimited number of attributes and associations in addition to those inherited from their parents. Attributes can be modeled with either simple or complex data types. To ensure semantic interoperability, the predefined types from CityGML or the standardized schemas of the ISO 19100 series of International Standards should be used wherever appropriate. This includes, amongst others, basic types from ISO/TS 19103, geometry and topology objects from ISO 19107, and temporal geometry and topology objects from ISO 19108.

If a predefined type is not available, ADEs can either define their own data types or import data types from external conceptual models. This explicitly includes the possibility of defining new geometry types not offered by ISO 19107. Designers of an ADE should however note that software might not be able to properly identify and consume such geometry types.

A feature type capturing a real-world feature with geometry should be derived either directly or indirectly from *Core::AbstractSpace* or *Core::AbstractSpaceBoundary*. By this means, the CityGML predefined spatial properties and the associated LOD concept are inherited and available for the feature type. If, however, these superclasses are either inappropriate or lack a spatial property required to represent the feature, an ADE may define new and additional spatial properties. If such a spatial property should belong to one of the predefined LODs, then the property name shall start with the prefix “*lodX*”, where *X* is to be replaced by an integer value between 0 and 3 indicating the target LOD. This enables software to derive the LOD of the geometry.

Constraints on model elements should be expressed using a formal language such as the Object Constraint Language (OCL). The ADE specifies the manner of application of constraints. However, following the CityGML conceptual model, constraints should at least be expressed on ADE subclasses of *Core::AbstractSpace* to limit the types of space boundaries (i.e., instances of *Core::AbstractSpaceBoundary*) that may be used to model the boundary of a space object.

9.3. Augmenting CityGML Feature Types with Additional ADE Properties

If a predefined CityGML feature type lacks one or more properties required for a specific application, a feasible solution in CityGML 2.0 was to derive a new ADE feature type as subclass of the CityGML class and to add the properties to this subclass. While conceptually clean, this

approach also faces drawbacks. If multiple ADEs require additional properties for the same CityGML feature type, this will lead to many subclasses of this feature type in different ADE namespaces. Information about the same real-world feature might therefore be spread over various instances of the different feature classes in an encoding making it difficult for software to consume the feature data.

For this reason, CityGML 3.0 provides a way to augment the predefined CityGML feature types with additional properties from the ADE domain without the need for subclassing. Each CityGML feature type has an extension attribute of name “adeOfFeatureTypeName” and type “ADEOfFeatureTypeName”, where *FeatureTypeName* is replaced by the class name in which the attribute is defined. For example, the *Building::Building* class offers the attribute *adeOfBuilding* of type *Building::ADEOfBuilding*. Each of these extension attributes can occur zero to unlimited times, and the attribute types are defined as abstract and empty data types.

If an ADE augments a specific CityGML feature type with additional ADE properties, the ADE shall create a subclass of the corresponding abstract data type associated with the feature class. This subclass shall also be defined as data type using the stereotype «*DataType*». The additional application-specific attributes and associations are then modeled as properties of the ADE subclass. This may include, amongst others, attributes with simple or complex data type, spatial properties or associations to other object and feature types from the ADE or external models such as CityGML.

The predefined “ADEOfFeatureTypeName” data types are called “hooks” because they are used as the head of a hierarchy of ADE subclasses attaching application-specific properties. When subclassing the “hook” of a specific CityGML feature type in an ADE, the properties defined in the subclass can be used for that feature type as well as for all directly or indirectly derived feature types, including feature types defined in the same or another ADE.

Multiple distinct ADEs can use the “hook” mechanism to define additional ADE properties for the same CityGML feature type. Since the “adeOfFeatureTypeName” attribute may occur multiple times, the various ADE properties can be exchanged as part of the same CityGML feature instance in an encoding. Software can therefore easily consume the default CityGML feature data plus the additional properties from the different ADEs.

Content from unknown or unsupported ADEs may be ignored by an application or service consuming an encoded CityGML model.

Designers of an ADE should favor using this “hook” mechanism over subclassing a CityGML feature type when possible. If an ADE must enable other ADEs to augment its own feature types (so-called ADE of an ADE), then it shall implement “hooks” for its feature types following the same schema and naming concept as in the CityGML conceptual model.

The UML fragment in Figure 86 shows an example for using the “hook” mechanism. For more details on this and other example ADEs, please see the [OGC CityGML 3.0 Users Guide](#).

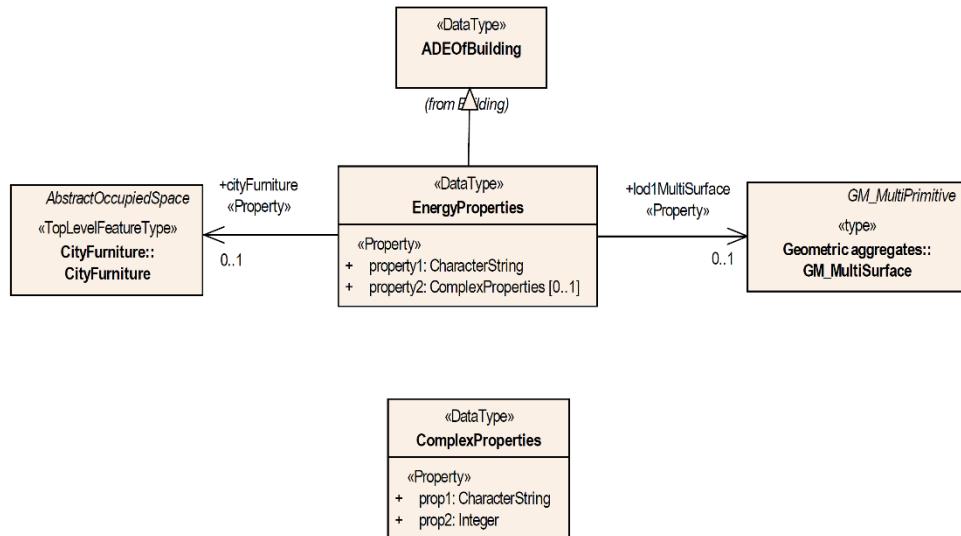


Figure 86 – The CityGML feature type Building is augmented with additional ADE properties by defining the data type EnergyProperties as a subclass of the ADE data type ADEOfBuilding.

9.4. Encoding of ADEs

This document only addresses the conceptual modeling of ADEs. Rules and constraints for mapping a conceptual ADE model to a target encoding are expected to be defined in a corresponding CityGML Encoding Standard. If supported, an ADE may provide additional mapping rules and constraints in conformance with a corresponding CityGML Encoding Standard.

9.5. Requirements and Recommendations

The following requirements and recommendations specify how ADEs shall be used as an extension capability to the CityGML Conceptual Model.

REQUIREMENTS CLASS 18

Target type	Conceptual Model
Dependency	/req/req-class-core

9.5.1. UML

Any extension to the CityGML Conceptual Model should be a faithful continuation of the styles and techniques used in that model. The following Requirements and Recommendations define a “faithful continuation”.

REQUIREMENT 48

An ADE SHALL be defined as conceptual model in UML in accordance with the conceptual modelling framework of the ISO 19100 series of International Standards

- | | |
|---|--|
| A | The UML model SHALL adhere to the General Feature Model as specified in ISO 19109. |
| B | The UML model SHALL adhere to rules and constraints for application schemas as specified in ISO/TS 19103. |
| C | Every ADE SHALL be organized into one or more UML packages having globally unique namespaces and containing all UML model elements defined by the ADE. |

RECOMMENDATION 1

In addition to meeting the requirements for a CityGML ADE, an ADE should:

- | | |
|---|---|
| A | The UML notations and stereotypes (Clause 5.2) used in the CityGML conceptual model SHOULD be applied to corresponding model elements in an ADE. |
| B | An ADE SHOULD import and use predefined classes from external conceptual UML models such as the CityGML modules or the standardized schemas of the ISO 19100 series of International Standards. |

9.5.2. Classes

The following Requirements and Recommendations define how CityGML classes should be extended by an ADE.

REQUIREMENT 49

ADEs typically extend CityGML by defining new Feature Types together with additional content such as Object Types, Data Types, Code Lists, and Enumerations.

- | | |
|---|---|
| A | Every Feature Type in an ADE SHALL be derived either directly or indirectly from the CityGML root Feature Type <i>core:AbstractFeature</i> or a subclass thereof. |
|---|---|

REQUIREMENT 49

- B UML classes representing Top-Level Feature Types SHALL use the «*TopLevelFeatureType*» stereotype.
- C Features representing spaces or space boundaries SHALL be derived either directly or indirectly from *core:AbstractSpace* or *core:AbstractSpaceBoundary* respectively.
- D An ADE may define new and additional spatial properties. If such a spatial property should belong to a predefined LOD, then the property name SHALL start with the prefix “*lodX*”, where X is an integer value indicating the target LOD.

RECOMMENDATION 2

ADEs typically extend CityGML by defining new feature types together with additional content such as object types, data types, code lists, and enumerations.

- A ADEs SHOULD use the predefined types from CityGML or the standardized schemas of the ISO 19100 series of International Standards.
- B Constraints on model elements SHOULD be expressed using a formal language such as the Object Constraint Language (OCL).
- C ADE subclasses of *core:AbstractSpace* SHOULD include constraints to limit the boundaries of the space object.

9.5.3. Properties

The following Requirements define how to use the CityGML extension properties to add attributes to an existing CityGML Feature Type.

REQUIREMENT 50

Every Feature Type includes an extension property (hook) of type “*ADEOf<FeatureTypeName>*” where *<FeatureTypeName>* is the name of that Feature Type. To add an extension property to a Feature Type:

- A The ADE SHALL create a subclass of the abstract data type associated with the hook.
- B This subclass SHALL be defined as a data type using the stereotype «*DataType*».
- C Application-specific attributes and associations SHALL be modeled as properties of the ADE subclass.



A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A.1. Introduction

CityGML 3.0 is a Conceptual Model. Since it is agnostic to implementing technologies, an Executable Test Script is not feasible. It becomes the responsibility of the Implementation Specifications to provide evidence of conformance. This evidence should be provided as an annex to the Implementation Specification document.

The test method specified in this ATS is manual inspection. Automated methods may be used where they exist.

A.2. Conformance Class Core

ABSTRACT TEST A.1

Subject	<xref target="req_core_classes" type="inline">/req/core/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Tunnel Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

ABSTRACT TEST A.1

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.2

Subject	/req/core/isorestrictions
Label	/ats/core/isorestrictions
Test purpose	To validate that none of the restrictions which the CityGML Conceptual Model imposes on ISO classes are violated by an Implementation Specification.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each instance of the GM_Solid class, validate that there are no interior boundaries associated with that instance.2. For each instance of a class descended from the GM_Solid class, validate that there are no interior boundaries associated with that instance.

ABSTRACT TEST A.3

Subject	<xref target="req_core_boundaries" type="inline">/req/core/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection

ABSTRACT TEST A.3

Test method

1. For each UML class defined or referenced in the Core Package:
 - a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.4

Subject

<xref target="req_core_ade_use" type="inline">/req/core/ade_use</xref>

Test purpose

To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.

Test method type

Manual Inspection

Test method

1. If any ADE classes or properties are included in the Core Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.3. Conformance Class Appearance

ABSTRACT TEST A.5

Subject

<xref target="req_appearance_classes" type="inline">/req/appearance/classes</xref>

Test purpose

To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

Test method type

Manual Inspection

Test method

1. For each UML class defined or referenced in the Appearance Package:
 - a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.
 - b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

ABSTRACT TEST A.5

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.6

Subject	<xref target="req_appearance_ade_use" type="inline">/req/appearance/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Appearance Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.4. Conformance Class CityFurniture

ABSTRACT TEST A.7

Subject	<xref target="req_cityfurniture_classes" type="inline">/req/cityfurniture/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

ABSTRACT TEST A.7

Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the City Furniture Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Modelf) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.8

Subject	<xref target="req_cityfurniture_boundaries" type="inline">/req/cityfurniture/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection

ABSTRACT TEST A.8

Test method

1. For each UML class defined or referenced in the City Furniture Package:
 - a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.9

Subject

<xref target="req_cityfurniture_ade_use" type="inline"/>/req/cityfurniture/ade_use</xref>

Test purpose

To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.

Test method type

Manual Inspection

Test method

1. If any ADE classes or properties are included in the City Furniture Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.5. Conformance Class CityObjectGroup

ABSTRACT TEST A.10

Subject

<xref target="req_cityobjectgroup_classes" type="inline"/>/req/cityobjectgroup/classes</xref>

Test purpose

To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

Test method type

Manual Inspection

Test method

1. For each UML class defined or referenced in the CityObject Group Package:
 - a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.
 - b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

ABSTRACT TEST A.10

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.11

Subject	<xref target="req_cityobjectgroup_boundaries" type="inline">/req/cityobjectgroup/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the CityObject Group Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.12

Subject	<xref target="req_cityobjectgroup_ade_use" type="inline">/req/cityobjectgroup/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection

ABSTRACT TEST A.12

Test method

1. If any ADE classes or properties are included in the City ObjectGroup Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.6. Conformance Class Dynamizer

ABSTRACT TEST A.13

Subject	<xref target="req_dynamizer_classes" type="inline">/req/dynamizer/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Dynamizer Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model

ABSTRACT TEST A.13

- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.14

Subject	<xref target="req_dynamizer_ade_use" type="inline">/req/dynamizer/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Dynamizer Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.7. Conformance Class Generics

ABSTRACT TEST A.15

Subject	<xref target="req_generics_classes" type="inline">/req/generics/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Generics Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition,

ABSTRACT TEST A.15

type, and multiplicity of those documented in the Conceptual Model.

- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.16

Subject	<xref target="req_generics_boundaries" type="inline"/>/req/generics/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Generics Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.17

Subject	<xref target="req_generics_use" type="inline"/>/req/generics/use</xref>
Test purpose	To validate that Generics are not used in a way that duplicates or conflicts with feature classes or attributes defined in the Conceptual Model..
Test method type	Manual Inspection

ABSTRACT TEST A.17

Test method

1. For all Generics-based classes and attributes defined in the Implementation Specification:
 - a) Demonstrate that this class or attribute does not duplicate or conflict with any classes or attributes defined in the Conceptual Model.

ABSTRACT TEST A.18

Subject

`<xref target="req_generics_ade_use" type="inline">/req/generics/ade_use</xref>`

Test purpose

To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.

Test method type

Manual Inspection

Test method

1. If any ADE classes or properties are included in the Generics Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.8. Conformance Class LandUse

ABSTRACT TEST A.19

Subject

`<xref target="req_landuse_classes" type="inline">/req/landuse/classes</xref>`

Test purpose

To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

Test method type

Manual Inspection

Test method

1. For each UML class defined or referenced in the LandUse Package:
 - a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.
 - b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

ABSTRACT TEST A.19

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.20

Subject	<xref target="req_landuse_ade_use" type="inline">/req/landuse/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the LandUse Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.9. Conformance Class PointCloud

ABSTRACT TEST A.21

Subject	<xref target="req_pointcloud_classes" type="inline">/req/pointcloud/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

ABSTRACT TEST A.21

Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the PointCloud Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Modelf) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.22

Subject	<xref target="req_pointcloud_ade_use" type="inline">/req/pointcloud/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection

ABSTRACT TEST A.22

Test method

1. If any ADE classes or properties are included in the Point Cloud Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.10. Conformance Class Relief

ABSTRACT TEST A.23

Subject

<xref target="req_relief_classes" type="inline">/req/relief/classes</xref>

Test purpose

To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

Test method type

Manual Inspection

1. For each UML class defined or referenced in the Relief Package:
 - a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.
 - b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.
 - c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
 - d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
 - e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model

Test method

ABSTRACT TEST A.23

- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.24

Subject	<xref target="req_relief_ade_use" type="inline">/req/relief/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Relief Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.11. Conformance Class Transportation

ABSTRACT TEST A.25

Subject	<xref target="req_transportation_classes" type="inline">/req/transportation/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Transportation Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition,

ABSTRACT TEST A.25

type, and multiplicity of those documented in the Conceptual Model.

- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.26

Subject	<xref target="req_transporation_boundaries" type="inline">/req/transportation/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ul style="list-style-type: none">1. For each UML class defined or referenced in the Transportation Package:<ul style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.27

Subject	<xref target="req_transporation_ade_use" type="inline">/req/transportation/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ul style="list-style-type: none">1. If any ADE classes or properties are included in the Transportation Package:<ul style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.12. Conformance Class Vegetation

ABSTRACT TEST A.28

Subject	<xref target="req_vegetation_classes" type="inline">/req/vegetation/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Vegetation Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Modelf) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model
Test method	

ABSTRACT TEST A.29

Subject	<xref target="req_vegetation_boundaries" type="inline">/req/vegetation/boundaries</xref>
----------------	--

ABSTRACT TEST A.29

Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Vegetation Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.30

Subject	<xref target="req_vegetation_ade_use" type="inline">/req/vegetation/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Vegetation Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.13. Conformance Class Versioning

ABSTRACT TEST A.31

Subject	<xref target="req_versioning_classes" type="inline">/req/versioning/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Versioning Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML

ABSTRACT TEST A.31

class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.32

Subject	<xref target="req_versioning_ade_use" type="inline">/req/versioning/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Versioning Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.14. Conformance Class WaterBody

ABSTRACT TEST A.33

Subject	<xref target="req_waterbody_classes" type="inline">/req/waterbody/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Waterbody Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Modelf) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model
Test method	

ABSTRACT TEST A.34

Subject	<xref target="req_waterbody_boundaries" type="inline">/req/waterbody/boundaries</xref>
----------------	--

ABSTRACT TEST A.34

Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Waterbody Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.35

Subject	<xref target="req_waterbody_ade_use" type="inline">/req/waterbody/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Waterbody Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.15. Conformance Class Construction

ABSTRACT TEST A.36

Subject	<xref target="req_construction_classes" type="inline">/req/construction/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Construction Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML

ABSTRACT TEST A.36

class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.37

Subject	<xref target="req_construction_boundaries" type="inline">/req/construction/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Construction Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.38

Subject	<xref target="req_construction_ade_use" type="inline">/req/construction/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.

ABSTRACT TEST A.38

Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Construction Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.16. Conformance Class Bridge

ABSTRACT TEST A.39

Subject	<xref target="req_bridge_classes" type="inline">/req/bridge/classes</xref>
Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Bridge Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles,

ABSTRACT TEST A.39

and multiplicity of those documented in the Conceptual Model

- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.40

Subject	<xref target="req_bridge_boundaries" type="inline">/req/bridge/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Bridge Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.41

Subject	<xref target="req_bridge_ade_use" type="inline">/req/bridge/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. If any ADE classes or properties are included in the Bridge Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.17. Conformance Class Building

ABSTRACT TEST A.42

Subject	<xref target="req_building_classes" type="inline">/req/building/classes</xref>
----------------	--

ABSTRACT TEST A.42

Test purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Building Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Modele) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Modelf) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.43

Subject	<xref target="req_building_boundaries" type="inline"/>/req/building/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection

ABSTRACT TEST A.43

Test method

1. For each UML class defined or referenced in the Building Package:
 - a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.44

Subject

[`<xref target="req_building_ade_use" type="inline">/req/building/ade_use</xref>`](#)

Test purpose

To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.

Test method type

Manual Inspection

Test method

1. If any ADE classes or properties are included in the Building Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.18. Conformance Class Tunnel

ABSTRACT TEST A.45

Subject

[`<xref target="req_tunnel_classes" type="inline">/req/tunnel/classes</xref>`](#)

Test purpose

To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model.

Test method type

Manual Inspection

Test method

1. For each UML class defined or referenced in the Tunnel Package:
 - a) Validate that the Implementation Specification contains a data element which represents the same concept as that defined for the UML class.
 - b) Validate that the data element has the same relationships with other elements as those defined for the UML class. Validate that those relationships have the same source, target, direction, roles, and multiplicities as those documented in the Conceptual Model.

ABSTRACT TEST A.45

- c) Validate that the data element has the same properties (attributes) as those specified for the UML class. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model.
- d) Validate that the properties of the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those properties have the same name, definition, type, and multiplicity of those documented in the Conceptual Model
- e) Validate that the associations represented for the data element include those of all superclasses of the UML class as documented in the Conceptual Model. Validate that those representations have the same source, target, roles, and multiplicity of those documented in the Conceptual Model
- f) Validate that the Implementation Specification enforces all constraints imposed on the UML class by the Conceptual Model

ABSTRACT TEST A.46

Subject	<xref target="req_tunnel_boundaries" type="inline">/req/tunnel/boundaries</xref>
Test purpose	To validate that the Implementation Specification does not specify boundaries except as defined in the Conceptual Model.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. For each UML class defined or referenced in the Tunnel Package:<ol style="list-style-type: none">a) Validate that the Implementation Specification does not specify boundaries for the UML class except as specified in the Conceptual Model.

ABSTRACT TEST A.47

Subject	<xref target="req_tunnel_ade_use" type="inline">/req/tunnel/ade_use</xref>
Test purpose	To validate that Application Data Extensions are not used unless conformance with the ADE Requirements Class can be demonstrated.
Test method type	Manual Inspection

ABSTRACT TEST A.47

Test method

1. If any ADE classes or properties are included in the Tunnel Package:
 - a) Validate that the Implementation Specification conforms with the ADE Requirements Class.

A.19. Conformance Class ADE

ABSTRACT TEST A.48

Subject

[`<xref target="req_ade_uml" type="inline">/req/ade/uml</xref>`](#)

Test purpose

To validate that Application Domain Extensions (ADE) to the City GML Conceptual Model are modeled correctly in UML.

Test method type

Manual Inspection

Test method

1. An ADE is defined as conceptual model in UML in accordance with the conceptual modeling framework of the ISO 19100 series of International Standards
 - a) Validate that the ADE UML model adheres to the General Feature Model as specified in ISO 19109.
 - b) Validate that the ADE UML model adheres to rules and constraints for application schemas as specified in ISO/TS 19103.
 - c) Validate that the ADE UML model is organized into one or more UML packages having globally unique namespaces and containing all UML model elements defined by the ADE.

ABSTRACT TEST A.49

Subject

[`<xref target="req_ade_elements" type="inline">/req/ade/elements</xref>`](#)

Test purpose

To validate that Application Domain Extension s (ADE) to the City GML Conceptual Model are implemented correctly.

Test method type

Manual Inspection

Test method

1. For each new UML class defined by an ADE:
 - a) Validate that every Feature Type class in an ADE is derived either directly or indirectly from the CityGML root Feature Type *core:AbstractFeature* or a subclass thereof.

ABSTRACT TEST A.49

- b) Validate that every UML class in an ADE which represents a top-level Feature Type is assigned the «*TopLevelFeature Type*» stereotype.
- c) Validate that every UML class in an ADE which represents spaces or space boundaries is derived either directly or indirectly from *core:AbstractSpace* or *core:AbstractSpace Boundary* respectively.
- d) Validate that any new or additional spatial properties defined by an ADE:
 1. belongs to a predefined LOD,
 2. has a property name which starts with the prefix “*lodX*”, where *X* is an integer value indicating the target LOD.

ABSTRACT TEST A.50

Subject	<xref target="req_ade_properties" type="inline">/req/ade/properties</xref>
Test purpose	To validate that Application Domain Extensions (ADE) to the CityGML Conceptual Model implement extension properties correctly.
Test method type	Manual Inspection
Test method	<ol style="list-style-type: none">1. Every Feature Type in the CityGML Conceptual Model includes an extension property whose purpose is to allow an ADE to add properties to that existing Feature Type. In every case where an extension property has been used:<ol style="list-style-type: none">a) Validate that the ADE creates a subclass of the abstract data type associated with the extension property.b) Validate that this subclass is defined as a data type using the stereotype «<i>DataType</i>».c) Validate that all application-specific attributes and associations for that Feature Type are modeled as properties of the ADE subclass.



B

ANNEX B (INFORMATIVE) GLOSSARY

ANNEX B (INFORMATIVE) GLOSSARY

B.1.

B.1.1. General

B.1.1.1. conformance test class

The set of conformance test modules that must be applied to receive a single certificate of conformance

[SOURCE: OGC 08-131, Clause 4.4]

B.1.1.2. feature

An abstraction of real world phenomena

[SOURCE: ISO 19101-1:2014, Clause 4.1.11]

B.1.1.3. feature attribute

A characteristic of a feature

[SOURCE: ISO 19101-1:2014, Clause 4.1.12]

B.1.1.4. feature type

A class of features having common characteristics

[SOURCE: ISO 19156:2011, Clause 4.7]

B.1.1.5. measurement

A set of operations having the object of determining the value of a quantity

[SOURCE: ISO 19101-2:2018, Clause 3.21]

B.1.1.6. model

An abstraction of some aspects of reality

[SOURCE: ISO 19109:2015, Clause 4.15]

B.1.1.7. observation

The act of measuring or otherwise determining the value of a property

[SOURCE: ISO 19156:2011, Clause 4.11]

B.1.1.8. observation procedure

A method, algorithm or instrument, or system of these, which may be used in making an observation

[SOURCE: ISO 19156:2011, Clause 4.12]

B.1.1.9. observation result

An estimate of the value of a property determined through a known observation procedure

[SOURCE: ISO 19156:2011, Clause 4.14]

B.1.1.10. property

A facet or attribute of an object referenced by a name.

[SOURCE: ISO 19143:2010, Clause 4.21]

B.1.1.11. requirements class

The aggregate of all requirement modules that must all be satisfied to satisfy a conformance test class

[SOURCE: OGC 08-131, Clause 4.19]

B.1.1.12. schema

The formal description of a model

[SOURCE: ISO 19101-1:2014, Clause 4.1.34]

B.1.1.13. sensor

A type of observation procedure that provides the estimated value of an observed property at its output

[SOURCE: OGC 08-094r1, Clause 4.5]

B.1.1.14. Standardization Target

An entity to which some requirements of a standard apply

[SOURCE: OGC 08-131, Clause 4.23]

B.1.1.15. timeseries

A sequence of data values which are ordered in time

[SOURCE: OGC 15-043r3]

B.1.1.16. universe of discourse

View of the real or hypothetical world that includes everything of interest

[SOURCE: ISO 19101-1:2014, Clause 4.1.38]

B.1.1.17. version

Particular variation of a spatial object

[SOURCE: INSPIRE Glossary]

B.1.2. ISO Concepts

The following concepts from the ISO TC211 Harmonized UML model are referenced by the CityGML Conceptual UML model but do not play a major role in its' definition. They are provided here to support a more complete understanding of the model.

B.1.2.1. Area

The measure of the physical extent of any topologically 2-D geometric object. Usually measured in “square” units of length.

[SOURCE: ISO 19103:2015]

B.1.2.2. Boolean

Boolean is the mathematical datatype associated with two-valued logic

[SOURCE: ISO 19103:2015]

B.1.2.3. CC_CoordinateOperation

A mathematical operation on coordinates that transforms or converts coordinates to another coordinate reference system.

[SOURCE: ISO 19111:2019]

B.1.2.4. Character

A symbol from a standard character-set.

[SOURCE: ISO 19103:2015]

B.1.2.5. CharacterString

CharacterString is a family of datatypes which represent strings of symbols from standard character-sets.

[SOURCE: ISO 19103:2015]

B.1.2.6. CRS

Coordinate reference system which is usually single but may be compound.

[SOURCE: ISO 19111:2019]

B.1.2.7. CV_DiscreteCoverage

A subclass of CV_Coverage that returns a single record of values for any direct position within a single geometric object in its spatiotemporal domain.

[SOURCE: ISO 19123:2005]

B.1.2.8. CV_DomainObject

An element of the domain of the CV_Coverage. It is an aggregation of objects that may include any combination of GM_Objects (ISO 19107:2003), TM_GeometricPrimitives (ISO 10108), or spatial or temporal objects defined in other standards, such as the CV_GridPoint defined in this International Standard.

[SOURCE: ISO 19123:2005]

B.1.2.9. CV_GridPointValuePair

A subtype of CV_GeometryValuePair that has a GM_GridPoint as the value of its geometry attribute.

[SOURCE: ISO 19123:2005]

B.1.2.10. CV_GridValuesMatrix

The geometry represented by the various offset vectors is in the image plane of the grid.

[SOURCE: ISO 19123:2005]

B.1.2.11. CV_ReferenceableGrid

A subclass of CV_Coverage that relates the grid coordinates to an external coordinate reference system.

[SOURCE: ISO 19123:2005]

B.1.2.12. Date

Date gives values for year, month and day. Representation of Date is specified in ISO 8601. Principles for date and time are further discussed in ISO 19108.

[SOURCE: ISO 19103:2015]

B.1.2.13. DateTime

A DateTime is a combination of a date and a time types. Representation of DateTime is specified in ISO 8601. Principles for date and time are further discussed in ISO 19108.

[SOURCE: ISO 19103:2015]

B.1.2.14. Distance

Used as a type for returning distances and possibly lengths.

[SOURCE: ISO 19103:2015]

B.1.2.15. EngineeringCRS

A contextually local coordinate reference system which can be divided into two broad categories:

1. earth-fixed systems applied to engineering activities on or near the surface of the earth;

2. CRSs on moving platforms such as road vehicles, vessels, aircraft or spacecraft.

[SOURCE: ISO 19111:2019]

B.1.2.16. FeatureType

metaclass that is instantiated as classes that represent individual feature types

[SOURCE: ISO 19109:2015, Clause 7.4.5]

B.1.2.17. GenericName

GenericName is the abstract class for all names in a NameSpace. Each instance of a GenericName is either a LocalName or a ScopedName.

[SOURCE: ISO 19103:2015]

B.1.2.18. Geometry

Geometry is the root class of the geometric object taxonomy and supports interfaces common to all geographically referenced geometric objects.

[SOURCE: ISO 19107:2003]

B.1.2.19. GM_CompositePoint

A GM_Complex containing one and only one GM_Point.

[SOURCE: ISO 19107:2003]

B.1.2.20. GM_CompositeSolid

A set of geometric solids adjoining one another along common boundary geometric surfaces

[SOURCE: ISO 19107:2003]

B.1.2.21. GM_GenericSurface

GM_Surface and GM_SurfacePatch both represent sections of surface geometry, and therefore share a number of operation signatures. These are defined in the interface class GM_GenericSurface.

[SOURCE: ISO 19107:2003]

B.1.2.22. GM_LineString

Consists of sequence of line segments, each having a parameterization like the one for GM_LineSegment

[SOURCE: ISO 19107:2003]

B.1.2.23. GM_MultiPrimitive

The root class for all primitive aggregates. The association role “element” shall be the set of GM_Primitives contained in this GM_MultiPrimitive. The attribute declaration here specializes the one at GM_Aggregate to include only GM_Primitives in this type of aggregate.

[SOURCE: ISO 19107:2003]

B.1.2.24. GM_OrientableSurface

A surface and an orientation inherited from GM_OrientablePrimitive. If the orientation is “+”, then the GM_OrientableSurface is a GM_Surface. If the orientation is “-”, then the

GM_OrientableSurface is a reference to a GM_Surface with an upNormal that reverses the direction for this GM_OrientableSurface, the sense of “the top of the surface”.

[SOURCE: ISO 19107:2003]

B.1.2.25. GM_PolyhedralSurface

A GM_Surface composed of polygon surfaces (GM_Polygon) connected along their common boundary curves.

[SOURCE: ISO 19107:2003]

B.1.2.26. GM_Position

A union type consisting of either a DirectPosition or of a reference to a GM_Point from which a DirectPosition shall be obtained.

[SOURCE: ISO 19107:2003]

B.1.2.27. GM_Primitive

The abstract root class of the geometric primitives. Its main purpose is to define the basic “boundary” operation that ties the primitives in each dimension together.

[SOURCE: ISO 19107:2003]

B.1.2.28. Integer

An exact integer value, with no fractional part.

[SOURCE: ISO 19103:2015]

B.1.2.29. Internet of Things

The network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the Internet.

B.1.2.30. IO_IdentifiedObjectBase

Supplementary identification and remarks information for a CRS or CRS-related object.

[SOURCE: ISO 19111:2019]

B.1.2.31. Length

The measure of distance as an integral, i.e., the limit of an infinite sum of distances between points on a curve.

[SOURCE: ISO 19103:2015]

B.1.2.32. Measure

The result from performing the act or process of ascertaining the extent, dimensions, or quantity of some entity.

[SOURCE: ISO 19103:2015]

B.1.2.33. Number

The base type for all number data, giving the basic algebraic operations.

[SOURCE: ISO 19103:2015]

B.1.2.34. Point

GM_Point is the basic data type for a geometric object consisting of one and only one point.

[SOURCE: ISO 19107:2003]

B.1.2.35. Real

The common binary Real finite implementation using base 2.

[SOURCE: ISO 19103:2015]

B.1.2.36. RS_ReferenceSystem

Description of a spatial and temporal reference system used by a dataset.

[SOURCE: ISO 19111:2019]

B.1.2.37. ScopedName

ScopedName is a composite of a LocalName for locating another NameSpace and a GenericName valid in that NameSpace. ScopedName contains a LocalName as head and a GenericName, which might be a LocalName or a ScopedName, as tail.

[SOURCE: ISO 19103:2015]

B.1.2.38. Solid

GM_Solid, a subclass of GM_Primitive, is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.

[SOURCE: ISO 19107:2003]

B.1.2.39. Time

Time is the designation of an instant on a selected time scale, astronomical or atomic. It is used in the sense of time of day.

[SOURCE: ISO 19103:2015]

B.1.2.40. TM_Duration

A data type to be used for describing length or distance in the temporal dimension.

[SOURCE: ISO 19108:2006]

B.1.2.41. TM_TemporalPosition

The position of a TM_Instant relative to a TM_ReferenceSystem.

[SOURCE: ISO 19108:2006]

B.1.2.42. UnitOfMeasure

Any of the systems devised to measure some physical quantity such distance or area or a system devised to measure such things as the passage of time.

[SOURCE: ISO 19103:2015]

B.1.2.43. URI

Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource

[SOURCE: ISO 19103:2015]

B.1.2.44. Volume

Volume is the measure of the physical space of any 3-D geometric object.

[SOURCE: ISO 19103:2015]



C

ANNEX C (INFORMATIVE) REVISION HISTORY

ANNEX C (INFORMATIVE) REVISION HISTORY

Table C.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2020-06-04	0.9.0	C. Heazel	all	Draft for review
2020-06-07	0.9.1	T. H. Kolbe	Chapter 10	Bibliography was added
2020-06-08	0.9.2	C. Nagel	Chapter 10	Chapter on ADE mechanism was added
2020-06-11	0.9.3	T. H. Kolbe	Chapter 7	Overview chapter on City GML was added
2020-06-11	0.9.4	T. Kutzner	Chapter 0	List of participants and submitters was added
2020-08-05	0.9.5	T. Kutzner	Chapter 8	Boundary constraints were added
2020-08-05	0.9.6	C. Heazel, T. Kutzner	Chapter 8	UML update



BIBLIOGRAPHY



BIBLIOGRAPHY

1. Patrick Cozzi, Sean Lilley, Gabby Getz: OGC 18-053r2, *OGC 3D Tiles Specification 1.0*. Open Geospatial Consortium (2019). <https://docs.ogc.org/cs/18-053r2/18-053r2.html>.
2. Kanishk Chaturvedi, Thomas H. Kolbe: OGC 16-098, *Future City Pilot 1 Engineering Report*. Open Geospatial Consortium (2017). <https://docs.ogc.org/per/16-098.html>.
3. Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, Karl-Heinz Häfele: OGC 12-019, *OGC City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium (2012). https://portal.ogc.org/files/?artifact_id=47842.
4. Carl Reed, Tamrat Belayneh: OGC 17-014r7, *OGC Indexed 3d Scene Layer (I3S) and Scene Layer Package Format Specification*. Open Geospatial Consortium (2020). <https://docs.ogc.org/cs/17-014r7/17-014r7.html>.
5. David Burggraf: OGC 12-007r2, *OGC KML 2.3*. Open Geospatial Consortium (2015). <https://docs.ogc.org/is/12-007r2/12-007r2.html>.
6. Steve Liang, Chih-Yuan Huang, Tania Khalafbeigi: OGC 15-078r6, *OGC SensorThings API Part 1: Sensing*. Open Geospatial Consortium (2016). <https://docs.ogc.org/is/15-078r6/15-078r6.html>.
7. Benjamin Hagedorn, Simon Thum, Thorsten Reitz, Voker Coors, Ralf Gutbell: OGC 15-001r4, *OGC® 3D Portrayal Service 1.0*. Open Geospatial Consortium (2017). <https://docs.ogc.org/is/15-001r4/15-001r4.html>.
8. Arne Bröring, Christoph Stasch, Johannes Echterhoff: OGC 12-006, *OGC® Sensor Observation Service Interface Standard*. Open Geospatial Consortium (2012). https://portal.ogc.org/files/?artifact_id=47599.
9. OASIS MQTT Technical Committee: *MQTT Version 5.0 Standard*, OASIS, March 7, 2019, Available from [OASIS](https://www.oasis-open.org/committees/tc_homepage.php?wg_abbrev=mqtt).
10. OGC: Definitions Register, Available from <http://www.opengis.net/def/>
11. OGC: Document Types Register, Available from <http://www.opengis.net/def/doc-type>
12. Open Geospatial Consortium: *The Specification Model – A Standard for Modular specifications*, [OGC 08-131](https://www.ogc.org/standards/08-131)
13. Object Management Group: Model Driven Architecture Guide rev. 2.0, Available from <https://www.omg.org/cgi-bin/doc?ormsc/14-06-01>
14. European Union: *INSPIRE Glossary*, Available from <https://inspire.ec.europa.eu/glossary>
15. Bhatia, S., Cozzi, P., Knyazev, A., Parisi, T.: *The GL Transmission Format (glTF)*, The Khronos Group, Available from <https://www.khronos.org/gltf>.

16. Elfes, A., 1989: *Using occupancy grids for mobile robot perception and navigation*. Computer 22(6):46–57. <https://doi.org/10.1109/2.30720>
17. Jensen, Christian S. and Dyreson, Curtis E.: *The Consensus Glossary of Temporal Database Concepts*. February 1998 Version. In: *Temporal Databases: Research and Practice* [online]. Springer Berlin Heidelberg, 1998. p. 367–405. Lecture Notes in Computer Science. Available from: 10.1007/BFb0053710
18. Snodgrass, Richard T: *Developing time-oriented database applications in SQL*. San Francisco, California : Morgan Kaufmann Publishers, July 1999. ISBN 1-55860-436-7. Available from: <http://www.cs.arizona.edu/~rts/tmdbbook.pdf>
19. Smith, B., Varzi, A. C., 2000: *Fiat and Bona Fide Boundaries*. Philosophy and Phenomenological Research, Vol. 60, No. 2, 401-420. <https://doi.org/10.2307/2653492>
20. Foley, J., van Dam, A., Feiner, S., Hughes, J., 2002: *Computer Graphics: Principles and Practice*. 2nd ed., Addison Wesley
21. Kolbe, T. H., Gröger, G., 2003: *Towards unified 3D city models*. In: Proceedings of the Joint ISPRS Commission IV Workshop on Challenges in Geospatial Analysis, Integration and Visualization II, Stuttgart, Germany. <https://mediatum.ub.tum.de/doc/1145769/>
22. Stadler, A., Kolbe, T. H., 2007: *Spatio-semantic Coherence in the Integration of 3D City Models*. In: Proceedings of the 5th International ISPRS Symposium on Spatial Data Quality ISSDQ 2007 in Enschede. http://www.isprs.org/proceedings/XXXVI/2-C43/Session1/paper_Stadler.pdf
23. Jensen, Christian S. and Snodgrass, Richard T., eds.: *TR-90, Temporal Database Entries for the Springer Encyclopedia of Database Systems*. Technical Report. TimeCenter, 22 May 2008. Available from: <http://timecenter.cs.aau.dk/TimeCenterPublications/TR-90.pdf>
24. Kolbe, T. H., 2009: *Representing and Exchanging 3D City Models with CityGML*. In: J. Lee, S. Zlatanova (Eds.), *3D Geo-Information Sciences, Selected Papers of the 3rd International Workshop on 3D Geo-Information in Seoul, Korea*. Springer, Berlin. https://doi.org/10.1007/978-3-540-87395-2_2
25. Johnson, Tom: *Bitemporal Data*. Elsevier, 2014. ISBN 978-0-12-408067-6. Available from: 10.1016/C2012-0-06609-4
26. Vretanos, P. A. 2010: *OpenGIS Web Feature Service 2.0 Interface Standard*, Open Geospatial Consortium. [OGC Doc. 09-025r1](http://www.opengeospatial.org/standards/wfs)
27. Becker, T., Nagel, C., Kolbe, T. H., 2011: *Integrated 3D Modeling of Multi-utility Networks and their Interdependencies for Critical Infrastructure Analysis*. In: T. H. Kolbe, G. König, C. Nagel (Eds.): *Advances in 3D Geoinformation Sciences*. LNG&C, Springer, Berlin. https://doi.org/10.1007/978-3-642-12670-3_1
28. Billen, R., Zaki, C. E., Servières, M., Moreau, G., Hallot, P., 2012: *Developing an ontology of space: Application to 3D city modeling*. In: Leduc, T., Moreau, G., Billen, R. (eds): *Usage, usability, and utility of 3D city models – European COST Action TU0801*, EDP Sciences, Nantes, Vol. 02007. <https://hal.archives-ouvertes.fr/hal-01521445>

29. Gröger, G., Plümer, L., 2012: *CityGML – Interoperable semantic 3D city models*. ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 71, July 2012. <https://dx.doi.org/10.1016/j.isprsjprs.2012.04.004>
30. Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015: *Applications of 3D City Models: State of the Art Review*. ISPRS International Journal of Geo-Information, 4(4). <https://doi.org/10.3390/ijgi4042842>
31. Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T., Kolbe, T. H., 2015: *Managing versions and history within semantic 3D city models for the next generation of CityGML*. In: Selected papers from the 10th International 3DGeoInfo Conference 2015 in Kuala Lumpur, Malaysia, Springer LNG&C, Berlin. https://doi.org/10.1007/978-3-319-25691-7_11
32. Nouvel, R., Bahu, J. M., Kaden, R., Kaempf, J., Cipriano, P., Lauster, M., Haefele, K.-H., Munoz, E., Tournaire, O., Casper, E., 2015: *Development of the CityGML Application Domain Extension Energy for Urban Energy Simulation*. In: Proceedings of Building Simulation 2015 – 14th Conference of the International Building Performance Simulation Association, IBPSA, 559-564. <http://www.ibpsa.org/proceedings/BS2015/p2863.pdf>
33. Chaturvedi, K., Kolbe, T. H., 2016: *Integrating Dynamic Data and Sensors with Semantic 3D City Models in the context of Smart Cities*. In: Proceedings of the 11th International 3D GeoInfo Conference, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-2/W1, ISPRS. <https://doi.org/10.5194/isprs-annals-IV-2-W1-31-2016>
34. Löwner, M.-O., Gröger, G., Benner, J., Biljecki, F., Nagel, C., 2016: *Proposal for a new LOD and multi-representation concept for CityGML*. In: Proceedings of the 11th 3D GeoInfo Conference 2016, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-2/W1, 3-12. <https://doi.org/10.5194/isprs-annals-IV-2-W1-3-2016>
35. Beil, C., Kolbe, T. H., 2017: *CityGML and the streets of New York – A proposal for detailed street space modeling*. In: Proceedings of the 12th International 3D GeoInfo Conference 2017, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-4/W5, ISPRS. <http://doi.org/10.5194/isprs-annals-IV-4-W5-9-2017>
36. Kaden, R., Clemen, C., 2017: *Applying Geodetic Coordinate Reference Systems within Building Information Modeling (BIM)*. In: Proceedings of the FIG Working Week 2017, Helsinki, Finland. https://www.fig.net/resources/proceedings/fig_proceedings/fig2017/papers/ts06h/TS06H_kaden_clemen_8967.pdf
37. Agugiaro, G., Benner, J., Cipriano, P., Nouvel, R., 2018: *The Energy Application Domain Extension for CityGML: enhancing interoperability for urban energy simulations*. Open Geospatial Data, Software and Standards, Vol. 3. <https://doi.org/10.1186/s40965-018-0042-y>
38. Biljecki, F., Kumar, K., Nagel, C., 2018: *CityGML Application Domain Extension (ADE): overview of developments*. Open Geospatial Data, Software and Standards, 3(1). <https://doi.org/10.1186/s40965-018-0055-6>

39. Konde, A., Tauscher, H., Biljecki, F., Crawford, J., 2018: *Floor plans in CityGML*. In: Proceedings of the 13th 3D GeoInfo Conference 2018, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-4/W6, 25-32, ISPRS. <https://doi.org/10.5194/isprs-annals-IV-4-W6-25-2018>
40. Kutzner, T., Hijazi, I., Kolbe, T. H., 2018: *Semantic modelling of 3D Multi-utility Networks for Urban Analyses and Simulations – The CityGML Utility Network ADE*. International Journal of 3-D Information Modeling (IJ3DIM) 7(2), 1-34. <https://dx.doi.org/10.4018/IJ3DIM.2018040101>
41. Labetski, A., van Gerwen, S., Tamminga, G., Ledoux, H., Stoter, J., 2018: *A proposal for an improved transportation model in CityGML*. In: Proceedings of the 13th 3D GeoInfo Conference 2018, ISPRS Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XLII-4/W10, 89–96. <https://doi.org/10.5194/isprs-archives-XLII-4-W10-89-2018>
42. Liu, Ling and Özsü, M. Tamer, eds.: *Encyclopedia of Database Systems*. New York, NY : Springer New York, 2018. ISBN 978-1-4614-8266-6. Available from: 10.1007/978-1-4614-8265-9
43. Chaturvedi, K., Kolbe, T. H., 2019: *A Requirement Analysis on Extending Semantic 3D City Models for Supporting Time-dependent Properties*. In: Proceedings of the 4th International Conference on Smart Data and Smart Cities, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. IV-4/W9, ISPRS. <https://doi.org/10.5194/isprs-annals-IV-4-W9-19-2019>
44. Kutzner, T., Chaturvedi, K. & Kolbe, T. H., 2020: *CityGML 3.0: New Functions Open Up New Applications*. PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science, 88, 43–61. <https://doi.org/10.1007/s41064-020-00095-z>