Masterarbeit

Entwicklung einer CityGML Application Domain Extension: Der Amtliche Lageplan in 3D

Wissenschaftliche Arbeit zur Erlangung des Grades
M.Sc. Geodäsie und Geoinformation
an der Fakultät für Bau Geo Umwelt der Technischen Universität München.

Betreut von	Dr.-Ing. Andreas Donaubauer (Lehrstuhl für Geoinformatik)
	Dr. rer. nat. Tatjana Kutzner (Lehrstuhl für Geoinformatik)
	Dr.-Ing. Andreas Rose (Vermessungsbüro Dr.-Ing. Andreas Rose
	Öffentlich bestellter Vermessungsingenieur)
	Lehrstuhl für Geoinformatik
Eingereicht von	Felix Müller
Eingereicht am	München, den 31.01.2021

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

München, 31.01.2021, Unterschrift

Abstract

From the continuously increasing degree of digitalization in all sectors of industry and society also demands and expectations on efficient processing of services in the public sector arise. Due to the future lack of qualified staff in public sectors this becomes a challenging task. Digitalization and automatization are considered to be an appropriate approach to reduce the workload of the workers, as rule-based and repetitive processes in the workflows can be outsourced to software components.

This thesis is embedded in the context of the automatization of processes regarding the building permit procedure.

In particular, the first part of the work is concerned about the development of a semantic data model of a 3D site plan as application domain extension (SiteplanADE) of the CityGML 3.0 model to replace the conventional CAD format of current site plans. The data model complements the CityGML 3.0 model with the conceptual domain as it provides classes for the modelling of objects from cadastre and spatial planning. To ensure interoperability and compatibility with respect to existing standards from the conceptual domain the respective models from INSPIRE and the Land Administration Domain Model are adopted and integrated on international level, as well as the models AAA and XPlanung on national level.

Besides the conventional use cases of a site plan also other use cases are considered in the site plan model, e.g. real estate valuation or estimation of insurance premiums.

The semantic properties of the site plan model make it possible to integrate automated processing steps already in the creation of a site plan, e.g. the computation of clearance spaces of the planned building.

In the second part of the thesis an algorithm is developed and implemented which computes the clearance spaces of a planned building model, based on further input parameters as a digital terrain model, the property parcel and spatial development areas. The algorithm implements the federal state building order of Nordrhein-Westfalen for the computation of clearance spaces.

To reduce the complexity and necessity of the general impression that is created by the building configuration application domain extensions for each CityGML 2.0 and CityGML 3.0 building models are created which provide the building model's components with attributes concerning their role and relevance in the context of the clearance space computation.

Finally, the results of the first two parts of the thesis are applied in an exemplary workflow for site plan generation which is implemented in FME. Starting with a conventional CAD site plan, the contained site plan objects are provided with 3D geometry by the use of digital terrain and surface models and mapped to the site plan model, before the objects are written out to a site plan file in the CityGML format.

Furthermore, aspects like schematron validation, electronic signature for the certification of CityGML site plan files, as well as further potentials of automatization are discussed.

Zusammenfassung

Im Zuge der fortschreitenden Digitalisierung in allen Bereichen der Gesellschaft und Industrie, steigen auch die Erwartungen und Anforderungen an die Effizienz von Verwaltungsprozessen. Aufgrund der hohen Erwartungshaltung bei der Verarbeitung von Anfragen und dem jetzt schon spürbaren Fachkräftemangel, wird in der Digitalisierung und Automatisierung regel-basierter Verwaltungsprozesse eine Möglichkeit gesehen, den Herausforderungen standzuhalten.
Diese Arbeit wird sich in den Kontext der Automatisierung von Prozessen rund um das Bauantragsverfahren einbetten.
Im Speziellen handelt der erste Teil der Arbeit von der Erstellung eines semantischen 3D-Datenmodells (*SiteplanADE*) als Erweiterung des CityGML 3.0-Modells, das die konventionellen amtlichen zwei-dimensionalen Lagepläne im Rahmen der Bauvorlagen des Bauantrags ersetzen soll.
Das Datenmodell ergänzt das CityGML 3.0-Modell mit der konzeptionellen Domäne im Kontext des (amtlichen) Lageplans und beinhaltet Objekte aus dem Kataster und der räumlichen Planung.
Um Interoperabilität und Kompatibilität zu existierenden Standards zu gewährleisten, werden Modelle von INSPIRE, Land Administration Domain Model auf internationaler Ebene, sowie AAA und XPlanung auf nationaler Ebene, im Lageplan-Modell nachempfunden. Neben klassischen Anwendungsfällen des Lageplans werden im Datenmodell des Lageplans auch andere Anwendungsfälle, wie beispielsweise die Bewertung von Immobilien und Abschätzung von Versicherungsbeiträgen berücksichtigt.
Aufgrund des semantischen Charakters des Lageplan-Modells können bereits während der Generierung eines Lageplans einige Arbeitsschritte, wie zum Beispiel die Berechnung von Abstandsflächen anhand des geplanten Gebäudemodells, automatisiert werden.
Ein Algorithmus, welcher basierend auf einem Geländemodell, Baugrundstück, Flächen des Baugebiets und eines Gebäudemodells im CityGML-Format die Abstandsflächen nach BauO NRW §6 berechnet, wird im zweiten Teil dieser Arbeit entwickelt. Aufgrund der Komplexität und Einfluss des Gesamteindrucks der Gebäude-Konfiguration müssen, um die Rolle und Relevanz von CityGML-Objekten in Bezug auf das Abstandsflächenrecht in den CityGML-Gebäudemodellen bereitstellen zu können, jeweils für CityGML 2.0 und CityGML 3.0 ADEs erstellt werden.
Schließlich wird mit den Ergebnissen der ersten beiden Teilen der Arbeit ein Workflow zur Erstellung eines Lageplans im CityGML-Format in FME realisiert. Ausgehend von einem konventionellen Lageplan im CAD-Format werden die enthaltenen Lageplan-Objekte anhand von digitalen Gelände- und Oberflächenmodellen mit 3D-Geometrie versehen, auf das Lageplan-Datenmodell (SiteplanADE) abgebildet und schließlich in eine Lageplan-Datei im CityGML-Format geschrieben. Des weiteren werden Aspekte wie Validierung mittels Schematron, die elektronische Signatur für die Beurkundung von CityGML-Lageplan-Dateien, sowie weitere Automatisierungsmöglichkeiten erörtert.

Abkürzungsverzeichnis

Zahlen

1D - ein-dimensional 121, 123
2D - zwei-dimensional ..... viii, xiv, 5, 27, 31, 34, 44, 46, 49, 51, 52, 57, 60, 85, 95, 123, 141, 158, 169–171, 178
3D - drei-dimensional ..... xii–xv, 3, 5, 7, 10–12, 16, 17, 26, 27, 29, 31, 34, 42, 44–46, 49, 51, 52, 57, 59, 60, 64, 78, 81, 83, 84, 141, 143, 144, 149, 151–153, 158, 161, 163, 169–172, 178–180, 182

A

AAA - ALFIS-ALKIS-ATKIS ..... vii, 7, 17, 28, 29, 31, 32, 34, 36, 42, 52, 59, 73, 167, 176
ADE - Application Domain Extension ... 7, 27–31, 33, 34, 36, 42, 53, 58, 60, 73, 83, 98, 101, 113–116, 170, 176, 177
AdV - Arbeitsgemeinschaft der Vemessungsverwaltungen der Länder der Bundesrepublik Deutschland 17, 29, 167
AEC - Architecture, Engineering and Construction 13, 26, 33, 49, 50
AFIS - Amtliches Festpunktinformationssystem 28, 88
ALKIS - Amtliches Liegenschaftskatasterinformationssystem .. xii, 17, 28, 36, 58, 67, 84, 88, 141, 145, 146, 182
API - Application Program Interface 44
ATKIS - Amtliches Topographisch-Kartographisches Informationssystem 28, 88

B

BauGB - Baugesetzbuch 22
BauO - Bauordnung 45, 73, 75, 77–79, 92, 95, 117, 134, 172, 173, 177, 178
BauPrüfVO - Verordnung über bautechnische Prüfungen 4, 5, 17–19, 28, 29, 88, 167
BDVI - Bund der Öffentlich bestellten Vermessungsingenieure ... vii, xii, 3, 20, 31, 32, 42, 68, 73, 142–144, 146
BGB - Bürgerliches Gesetzbuch 40
BIM - Building Information Modeling ..... vii, 3, 10, 11, 14, 16, 27, 28, 30, 32, 33, 81, 175
BMZ - Baumassenzahl 21, 34, 37, 42, 45, 46
BPMN - Business Process Model and Notation 2, 16, 180, 183
B-Rep - Boundary Representation 81, 175
bspw. - beispielsweise .. 25, 27, 32, 37, 43–46, 51, 56–58, 60, 67, 73, 74, 86, 101, 104, 112, 115, 116, 118, 121, 123, 141, 144, 159, 161, 167, 169, 170, 177, 180
bzgl. - bezüglich .. 6, 10, 11, 16, 25, 27, 29, 31, 42, 49–51, 54, 60, 69, 78, 79, 81, 85–89, 92, 95, 98, 103, 109, 116, 127, 147, 156, 169, 174, 176, 177, 180, 182
bzw. - beziehungsweise x, xiv, 2, 7, 28, 30, 35, 36, 44, 45, 49–52, 58, 60, 70, 73, 75–77, 81, 82, 87, 88, 91, 92, 96, 98, 99, 103, 104, 141, 148, 156, 167, 169, 172, 173

C

CA - Certificate Authority 39, 40
ca. - circa 89, 94, 160
CAD - computer-aided design ..... xv, 3, 141, 143–146, 158, 178
CityGML - City Geographic Markup Language ..... vii, x, xiii, xv, 7, 26–36, 42–44, 50–53, 57–60, 62, 65–71, 73, 74, 79–85, 96–104, 109, 113–116, 118, 119, 123, 141, 143, 144, 146, 156–158, 161, 167, 169, 170, 172, 175–180
CRS - Coordinate Reference System 88, 89, 92
CSG - Constructed Solid Geometry 81, 175

D

DGM - Digitales Geländemodell. ix, xiii, 7, 25, 26, 28, 34, 42, 76, 78, 86–89, 94, 96, 99, 100, 123, 125, 139, 141, 144–147, 149–151, 153–155, 159, 160, 172, 178, 182
d.h. - das heißt 5, 11, 28, 35, 46, 58, 80–82, 89, 92, 132, 173
DLKM - Digitales Liegenschaftskataster-Modell 17
DMN - Decision Model and Notation 2, 16, 180
DOM - Document Object Model 103, 122, 123, 129, 161, 178
DOM* - Digitales Oberflächenmodell ..... xiii, 141, 144–146, 150, 151, 154–156, 178, 182
DOP - Digitales Ortho-Photo ..... xii, 141, 145, 146

E

EGFOK - Ergeschoss-Fußbodenoberkante 23
etc. - et cetera 7, 10, 16, 25, 33–35, 37, 43, 49, 51, 57, 61, 64, 73, 78, 79, 101, 141
ETRS89 - European Terrestrial Reference System 1989 88
EU - Europäische Union 36
EUnet4DBP - European Network for Digital Building Permits 13
EuroSDR - European Spatial Data Research 32

F

FME - Feature Manipulation Engine . xii–xiv, 7, 44, 59, 72, 99, 141, 144–148, 156–158, 161, 170, 178–180, 182

G

GCG2016 - German Combined QuasiGeoid 2016 88
GFZ - Geschossflächenzahl 23, 34, 37, 45, 46
ggf. - gegebenenfalls 8, 25, 98
GIS - Geoinforamtionssystem ..... vii, 27, 30, 33, 49, 50, 53, 74, 141, 143, 175, 178
GML - Geographic Markup Language ... 29, 33, 34, 50, 103, 109, 148, 153–159, 161, 179
GRZ - Grundflächenzahl 23, 34, 37, 45, 46

H

HTML - Hypertext Markup Language ..... x, 104, 106, 108, 112, 161

I

ID - Identifikator 148, 153–155, 157

IFC - Industry Foundation Classes 14, 27, 28, 30–32, 81
INSPIRE - Infrastructure for Spatial Information in Europe .... viii, ix, 7, 28, 34–36, 52, 58, 59, 62, 63, 66–69, 71–73, 167, 176
IoT - Internet of Things 43
ISO - International Organization for Standardization 62

L

LADM - Land Administration Domain Model .. 7, 27, 34–36, 52, 58, 59, 66, 70, 71, 73, 176
LCDM - Legal Cadastral Domain Model 35
LOD - Level of Detail .... xiv, xv, 4, 6, 27, 29, 31, 44, 57, 64, 82–84, 141, 145, 146, 158, 169, 170, 178
LOD* - Level of Development 10

M

MB - Megabyte 158

N

NRW - Nordrhein-Westfalen . 4, 5, 17–19, 29, 45, 50, 75, 77–79, 88, 92, 95, 173, 177, 178

O

o.ä. - oder Ähnliches 16, 26, 60, 62, 116
OCL - Object Constraint Language ..... x, 44, 61, 83, 98, 103, 109, 115
ÖBVI - Öffentlich bestellter Vermessungsingenieur 5, 10
ÖPNV - Öffentlicher Personennahverkehr 43
ÖREB - Öffentlich-rechtliche Eigentumsbeschränkung 35
OGC - Open Geospatial Consortium 27, 33

R

RRR - Right-Restriction-Responsibility 35
RSA - Rivest–Shamir–Adleman 40

S

SIG3D - Special Interest Group 3D 83
SigG - Signaturgesetz 37
SVRL - Schematron Validation Report Language 104, 108, 110

T

TIC - Terrain Intersection Curve 100, 125
TIN - Triangulated Irregular Network ..... xiii, 144, 145, 154, 155

U

u.a. - unter anderem 8, 10, 13, 35, 43, 45, 66, 73, 95, 129, 165, 167
u.ä. - und Ähnliches 11
UAV - Unmanned Aerial Vehicle 86

UML - Unified Modeling Language ..... xii, 33, 44, 46, 83, 103, 129
usw. - und so weiter 65, 95
UTM - Universal Transverse Mercator ..... ix, x, 85, 88, 89, 91–94

V

v.a. - vor allem 1, 31, 33, 34, 53, 60, 68, 81, 87, 117, 159, 169, 173, 174, 180
VermKatG - Vermessungs- und Katastergesetz 50, 176
vgl. - vergleiche 70, 103, 116–118, 123, 160, 164, 176, 180, 182

W

W3C - World Wide Web Consortium 6, 41, 164
WFS - Web Feature Service 182

X

XAdES - XML Advanced Electronic Signatures 6, 41
XML - Extensible Markup Language .. viii, x, xiv, 6, 29, 33, 37, 41, 44, 61, 62, 68, 72, 74, 95, 101, 103, 104, 108, 115, 129, 130, 153, 163–165, 171, 180, 181
XMLDSig - XML Digital Signature 6, 41
XSD - XML Schema Definition 33, 46, 102, 115, 180
XSL - Extensible Stylesheet Language 61, 104
XSLT - Extensible Stylesheet Language Transformation 61, 104, 108, 180

Z

z.B. - zum Beispiel .. xiii, 3, 4, 8, 10, 16, 25–28, 31, 33–35, 37, 42, 43, 45, 46, 51, 58, 60, 61, 63–67, 73, 77, 81, 82, 84–86, 92, 101, 104, 127, 141, 144, 146, 150, 158, 163, 170, 172, 175–177, 180, 183
ZPO - Zivilprozessordnung 171
z.T. - zum Teil 146

Abbildungsverzeichnis

Abbildung 1 Tätigkeitsbereiche, die Potenzial für Automatisierung bieten. 2

Abbildung 2 Automatisierbare Arbeitszeit nach Berufsgruppen im öffentlichen Dienst. . 2

Abbildung 3 Schema des Ablaufs im Baugenehmigungsverfahren (Landeshauptstadt München Referat für Statdplanung und Bauordnung Lokalbaukomission, 2017). 9

Abbildung 4 Workflow für das Baugenehmigungsverfahren unter Anwendung des GeoBIM-Ansatzes (Noardo et al., 2019a). 11

Abbildung 5 Anwendungsfall des Lageplans im Prozess der Bauwerksplanung auf Basis des Lageplans. 12

Abbildung 6 Vereinfachter Workflow für die Bauplanung mit GeoBIM (Noardo et al., 2019a). 12

Abbildung 7 Workflow des BIM-basierten Bauantrags (Theiler, 2020a). 15

Abbildung 8 Eingangsdaten bei der Erstellung eines Lageplans. 26

Abbildung 9 Unterschiede der Skalenbereiche zwischen GIS (Geoinforamtionssystem) und BIM (Building Information Modeling) (Kaden, Seuß & Kolbe, 2019). 30

Abbildung 10 Thematische Abdeckung der Datenmodelle AAA (ALFIS-ALKIS-ATKIS)-Fachschema und CityGML (City Geographic Markup Language) bezogen auf den Objektkatalog des BDVI (Bund der Öffentlich bestellten Vermessungsingenieure)s. 32

Abbildung 11 Modellierungslayer von internationaler Ebene zu nationaler Ebene. Die verschiedenen Standards, welche für das Lageplan-Modell fusioniert werden, weisen bereits wechselseitige Einflüsse und Beziehungen untereinander auf. 36

Abbildung 12 Die digitale Signatur dient der Authentizität durch Signieren, sowie dem Schutz der Daten durch Verschlüsseln (Bitzer & Brisch, 1999). 38

Abbildung 13 Die digitale Signatur dient der Authentizität durch Signieren, sowie dem Schutz der Daten durch Verschlüsseln (Elektronik-Kompendium.de, 2021). 39

Abbildung 14 Ablauf der Signatur eines Dokuments (links) und resultierende Datei mit Originaldokument, Hash-Wert und Zertifikat (rechts) (Bitzer & Brisch, 1999). 39
Abbildung 15 Validierung des empfangenen Dokuments anhand des Hash-Werts (Paganini, 2013). 40
Abbildung 16 Abstufung verschiedener elektronischer Signaturen (Hühnlein & Korte, 2006). 41
Abbildung 17 Unterschiedliche Realisierungen der XML (Extensible Markup Language)-Signatur (Hühnlein & Korte, 2006). 41
Abbildung 18 Verschiedene topologische Relationen in 2D (zwei-dimensional) (Gröger & George, 2012). 46
Abbildung 19 Use Case Diagramm für den Anwendungsfall des Lageplans. 48
Abbildung 20 Schematische Darstellung des einfachen Schreibens in ein neues Datenformat bei Kompatibilität des Eingangs- und Ausgangsdatenmodell. Die Abbildung ist sinngemäß zu verstehen und hat die Reader/Writer-Funktionalität der Safe Software FME zur Vorlage. 59
Abbildung 21 Darstellung von möglichen näheren Spezifikationen von thematischen Begrenzungsflächen eines Gebäudes. 61
Abbildung 22 Schritte bei der Validierung eines XML-Instanz-Dokuments mittels Schematron (Robertson, 2003). 62
Abbildung 23 Visualisierung der INSPIRE (Infrastructure for Spatial Information in Europe)-Thematik *Natural risk zones* (*D2.8.III.12 Data Specification on Natural Risk Zones – Technical Guidelines*, 2013) 63
Abbildung 24 Beispielhafte Visualisierung eines Konflikts, bei welcher die maximal zulässige Gebäudehöhe überschritten wird (Emamgholian, Pouliot & Shojaei, 2020) 65
Abbildung 25 Package-Diagramm des Pakets LandUsePlanning. 66
Abbildung 26 Auszug aus einem Klassendiagramm des Subpackages *LandUsePlanningXPlanung*. Die Klasse *BP_BaugebietsTeilflaeche* erbt sowohl von den Klassen *BP_ZusaetzlicheFestsetzungen, BP_GestaltungBaugebiet, BP_Objekt*, sowie der INSPIRE-Verknüpfungsklasse *ZoningElement*. 69

Abbildung 27 3D-Objekte eines Raumplanungswerk, welches Baugebiete mit maximaler Gebäudehöhe mittels Solid-Geometrien implementiert (Bydłosz, Bieda & Parzych, 2018). 70
Abbildung 28 Package-Diagramm des Pakets LandAdministrationCadastre. 70
Abbildung 29 Schematische Visualisierung des Bodenprofil-Konzeptes aus der INSPIRE-Spezifikation (*D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines*, 2013). 72
Abbildung 30 Der Bodenkörper besteht aus einer Anzahl abgeleiteter Bodenprofile (*D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines*, 2013). 72
Abbildung 31 Darstellung der Abstandsflächen, welche sich aus den einzelnen Gebäudekomponenten ergeben, sowie des resultierenden Abstandsraumes. 74
Abbildung 32 Darstellung der Parameter in Gleichung 4.1 für die Abstandsflächentiefenberechnung. 77
Abbildung 33 Darstellung der in CityGML zusammengefassten kohärenten Modellierung von Semantik und Geometrie (Stadler & Kolbe, 2007). 79
Abbildung 34 Darstellung der Schnittlinie *SL* des oberen Wandabschluss mit der Dachhaut (Noack et al., 2005). 80
Abbildung 35 Die thematischen Flächen *RoofSurface* und *WallSurface* begrenzen den Volumenkörper (*Solid*) des Gebäudemodells (Special Interest Group 3D, 2018d). 80
Abbildung 36 Lineare Änderung dT der Abstandsflächentiefe in Abhängigkeit der Summe der Komponenten des Verschiebungsvektors $\mathbf{dx}$. 87
Abbildung 37 Darstellung der fehlerhaften Bestimmung der Abstandsflächentiefe bei Verwendung eines zu grob-auflösenden DGM (Digitales Geländemodell)s. 88
Abbildung 38 Darstellung der Zylinderabbildung bei der UTM (Universal Transverse Mercator)-Projektion (Aumann et al., 2016). 89
Abbildung 39 Verzerrung (bei Strecken auf dem Ellipsoid) pro Kilometer in Abhängigkeit des Abstands zum Hauptmeridian. 90
Abbildung 40 Visualisierung der verschiedenen Bezugssysteme der Strecken und deren Reduktionsbeziehungen untereinander (*Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten*, 2019). 90

Abbildung 41 Streckenverzerrung bei der Abbildung der Strecken von Ellipsoid/Geländeoberfläche auf die Projektionsebene der UTM-Projektion (Kaden, 2016). 91
Abbildung 42 Über die Lagekoordinaten der vertikalen Wandbegrenzungen P_0 und P_1 werden normierte Richtungsvektoren $\mathbf{r}_{01}$ und senkrecht dazu $\mathbf{r}_{\perp 01}$ berechnet. Diese werden mit den Faktoren der Wandlänge L bzw. (beziehungsweise) der Abstandsflächentiefe T skaliert. Mittels der Geradengleichung $P_{neu} = P_{Aufpunkt} + \lambda \cdot \mathbf{r}_{Richtung}$ werden dann die Knotenpunkte des Abstandsflächenpolygons berechnet. 92
Abbildung 43 Differenz zwischen der örtlichen Strecke $S_H = 10\,m$ und der projizierten Strecke im UTM-System in Abhängigkeit des Meridianabstands und der ellipsoidischen Höhe mit mittlerem Krümmungshalbmesser $R = 6381\,km$ für mittlere Breiten Deutschlands. 93
Abbildung 44 Verzerrung der Strecke bei der Reduktion von der örtlichen Strecke mit ellipsoidischen Höhen auf das Referenzellipsoid. 93
Abbildung 45 Darstellung der Größen für die Abschätzung des maximalen Abstands zum Hauptmeridian für einen bestimmten Breitengrad. 94
Abbildung 46 Module für das Eingangsdatenprofil für CityGML 2.0. 97
Abbildung 47 Module für das Eingangsdatenprofil für CityGML 3.0. 98
Abbildung 48 Die Prozesse P und ihre sequentielle Abfolge wird in Blöcken von Quellcode implementiert..... 99
Abbildung 49 Aktivitätsdiagramm der automatisierten Abstandsflächenberechnung. 100
Abbildung 50 Klassendiagramm des Gebäude-Profils in CityGML 2.0. 102
Abbildung 51 Anzeige des fehlerhaften Elements im HTML (Hypertext Markup Language)-Validierungsbericht.....108
Abbildung 52 Klassendiagramm des CityGML 3.0-Gebäudemodells mit OCL (Object Constraint Language)-Bedingungen. 109
Abbildung 53 Inspektion des fehlerhaften Kind-Element des *Root*-Elements *CityModel* im HTML-Report. 112
Abbildung 54 Zwei-phasiger Validierungsprozess eines XML-Dokuments mit XML-Schema- und Schematron-Validierung. 115

Abbildung 55 Gesimse sollen nicht als Teil der Außenwand, sondern als dekorative Bauelemente, also als Gebäudeinstallationen, modelliert werden (Poellet, 2008). 116
Abbildung 56 Draufsicht auf ein Gebäude mit Balkon. Ein Balkon der sich über zwei Gebäudeseiten W_0 und W_1 erstreckt (links), soll in zwei individuelle Balkone unterteilt werden, um diese mittels einer 1:1-Beziehung mit den Gebäudeseiten referenzieren zu können (rechts). 116
Abbildung 57 Der Balkon grenzt an die Beiden Außenwände W_0 und W_1 an. Das dominierende Referenzobjekt soll über die Klasse *CityObjectRelation* definert werden. 117
Abbildung 58 Klassendiagramm für die Programm-interne Repräsentation der thematischen Teilflächenklassen. 119
Abbildung 59 Klassendiagramm für die Programm-interne Repräsentation der Container-Klassen. 120
Abbildung 60 Klassendiagramm für die Programm-interne Repräsentation der Klassen für Gebäude und Gebäudeteile. 120
Abbildung 61 Klassendiagramm für die Programm-interne Repräsentation der Klassen für Baugebietsflächen und des Gebäudegrundstücks. 121
Abbildung 62 Dekomposition eines Gebäudemodells (*a)*) in seine Komponenten (*b)*) und Erstellung eines Komponentengraphs (*c)*). In grün ist beispielhaft ein Durchlauf durch den Graph von einer Dachkomponente zur Bodenkomponente (oder andersherum) dargestellt. 122
Abbildung 63 Kontext des Prozesses *Vorverarbeitung* innerhalb der Aktivität. 122
Abbildung 64 Aktivitätsdiagramm des Prozesses *Vorverarbeitung*. 124
Abbildung 65 Kontext des Prozesses *Untersuchung auf Abstandsflächen* innerhalb der Aktivität. 125
Abbildung 66 Aktivitätsdiagramm des Prozesses *Untersuchung auf Abstandsflächen*. .. 126
Abbildung 67 Kontext des Prozesses *Prüfung, ob Abstandsfläche erforderlich* innerhalb der Aktivität. 126
Abbildung 68 Aktivitätsdiagramm des Prozesses *Prüfung, ob Abstandsfläche erforderlich*. 128

Abbildung 69 UML (Unified Modeling Language)-Klassendiagramm des Entscheidungsbaums. 129
Abbildung 70 Kontext des Prozesses *Berechnung der Abstandsflächen* innerhalb der Aktivität. 132
Abbildung 71 Aktivitätsdiagramm des Prozesses *Berechnung der Abstandsflächen*. 133
Abbildung 72 Aktuelle Teilfläche (orange) innerhalb der konvexen Hülle (rot) einer anderen zur Seite geöffneten konkaven Teilfläche (grün). 133
Abbildung 73 Aktuelle Teilfläche (orange) innerhalb der konvexen Hülle (rot) einer anderen nach oben geöffneten konkaven Teilfläche (grün). 134
Abbildung 74 Darstellung der Konstruktion der Abstandsflächen für ein Liniensegment einer Teilfläche. 135
Abbildung 75 Anwendung des Algorithmus zur automatischen Berechnung von Abstandsflächen auf verschiedene Gebäudemodelle. 140
Abbildung 76 Workflow für die Erstellung des 3D-Lageplans. 142
Abbildung 77 Ausschnitt aus dem vom BDVI zur Verfügung gestellten Lageplan. 142
Abbildung 78 Unter zu Hilfenahme eines DOP (Digitales Ortho-Photo) werden zusätzlich die nicht im Lageplan enthaltenen Landbedeckungsflächen digitalisiert. Flächen des geplanten Gebäudes, sowie der Gebäude-Footprints werden mittels entsprechender (ALKIS (Amtliches Liegenschaftskatasterinformationssystem)-)Shape-Dateien ausgespart. 145
Abbildung 79 Ausschnitt aus der FME (Feature Manipulation Engine)-Workbench. Zu sehen sind die Eingangsdatenströme in den *PythonCaller*, in welchem die Abstandsflächenberechnung durchgeführt wird. 147
Abbildung 80 Ausschnitt aus der FME-Workbench. Die aus dem *PythonCaller* resultierenden *Features* werden basierend auf dem ”clearanceSpaceType” gefiltert und weiterverarbeitet. 148
Abbildung 81 Über den Transformer *FeatureMerger* werden die Abstandsflächen über die Attribute ”parent_id” mit den zugehörigen Gebäudekomponenten und deren Attribut ”gml_id” referenziert. 149
Abbildung 82 Filterung der *Features* basierend auf dem Attribut zur Gewinnung von 3D (drei-dimensional)-Information. 149

Abbildung 83 Darstellung des Unterschieds zwischen DGM (graue Linie) und DOM* (Digitales Oberflächenmodell) (blaue Linie) (Waser, 2020). 150
Abbildung 84 Darstellung des Unterschieds zwischen DGM (graue Linie) und DOM* (blaue Linie) (Waser, 2020). 150
Abbildung 85 Die *Feature*-Geometrie wird um die z-Komponente ergänzt. Der Wert wird aus dem Text-Attribut extrahiert, welches den Höhenwert beinhaltet. 151
Abbildung 86 Referenzierung der *Features* mit den korrespondieren 3D-Bibliothek-Modellen. 152
Abbildung 87 Referenzierung der *Features* werden mit der Instanz-Geometrie. 153
Abbildung 88 Drapieren der Liniensegmente über die TIN (Triangulated Irregular Network)-Oberfläche. 154
Abbildung 89 Filterung der DGM-Punkte und Zuschneiden der entstehenden TIN-Oberfläche. 155
Abbildung 90 Zusammenfügung der zusammengehörenden Oberflächenbestandteile gemäß des für die Schneide-Operation verwendeten *Features*. 155
Abbildung 91 Extraktion von Oberflächen aus dem Punktwolken-DOM*. 156
Abbildung 92 *Feature Type*-Reiter: *User Attributes* und *Format Attributes*. 157
Abbildung 93 CityGML-Profil, welches die Lageplan-Objekte des erstellen Lageplans im CityGML-Format. 158
Abbildung 94 Fläche mit DGM mit hoher Punktdichte (links) und Fläche mit zwei DGMs unterschiedlicher Punktdichte (rechts). 160
Abbildung 95 Inspektion des 3D-Lageplans im *FME Data Inspector*. In dem Panel links sind die eingelesenen Objektarten (*Feature Types*) zu sehen. Unten in der Tabelle sind die enthaltenen Daten der einzelnen Objekte (*Features*) tabellarisch aufgelistet. Im rechten Bereich sind zusätzlich FME-interne Attribute zu sehen, die bei der Abbildung auf die FME-interne Darstellung entstehen. 161
Abbildung 96 Visualisierung des Lageplans im Browser. Über JavaScript-Programmierung können die 3D-Objekte dynamisch manipuliert werden, wie hier z.B. (zum Beispiel) das Ein- und Ausblenden über das Setzen von Häkchen in der Legende. 163

Abbildung 97 Die Einteilung der Lageplan-Objekte in die topographische und konzeptionelle Domäne, sowie in deren Zustand (existent oder geplant), resultiert in vier Schnittmengen. 168
Abbildung 98 Abnahme des benötigten LOD (Level of Detail)s in Abhängigkeit zur Distanz zum Bauprojekt bzw. zum Objekt des Interesses (Quelle des Gebäudes in verschiedenen LODs: (Kolbe et al., 2020)). 169
Abbildung 99 Die 3D-Geometrie des Gebäude-Objekts wird in der FME dereferenziert und in eine 2D-Geometrie transformiert. 170
Abbildung 100 Das Gebäude-Objekt wurde mit 3D- und 2D-Geometrie instanziiert. Für die Erstellung eines 2D-Lageplans wird in einer Software die 2D-Geometrie extrahiert. 171
Abbildung 101 Gebäudemodell, mit detaillierter Balkonmodellierung (oben links), die resultierenden Abstandsflächen für jede einzelne Fläche der Balkongeländer (oben rechts) und die resultierenden Abstandsflächen für die geometrische Hüllen der Balkongeländer (unten Mitte). 174
Abbildung 102 Generalisierte Modellierung der Balkongeländer durch geometrische Hüllen der Einzelflächen (grün). 175
Abbildung 103 Schematische Darstellung einer XProc-Pipeline für die Verarbeitung zweier Eingangsdokumente, und der Ausgabe des Ergebnisdokuments (Siegel, 2019). 181
Abbildung 104 In einem Anwendungsfall des Lageplans wird ein Workflow-Diagramm für die Prozessierungsabfolge erstellt. Die Validierung des XML-Dokuments wird in diesen Workflow als Prozess definiert, der mittels XProc die Gültigkeit des Dokuments im Kontext des Anwendungsfalls überprüft, bevor es in den restlichen Teil des Workflows zusammen mit den anderen Eingangsdaten verarbeitet wird. 181
Abbildung 105 Schematische Darstellung der Vollautomatisierten Erstellung eines Lageplans. 182

Tabellenverzeichnis

Tabelle 1 Abdeckung der Eingangsdaten über Objektklassen verschiedener Datenmodelle, welche in das Datenmodell des 3D-Lageplans übernommen werden. 42

Tabelle 2 Innovationen und deren Relevanz für die Erstellung eines Lageplans. 43

Tabelle 3 Anwendungsfelder des (amtlichen) Lageplans und deren Anforderungen an das 3D-Lageplan-Datenmodell. 49

Tabelle 4 Thematische Aufteilung der Objektmenge im Kontext des Lageplans. 53

Tabelle 5 Vorschläge thematischer Gruppierungen in *CityObjectGroups* und deren Anwendung. 56

Tabelle 6 Vorgesehene Geometrietypen für die verschiedenen Lageplan-Objekte. 57

Tabelle 7 LOD0-4 des CityGML 2.0 Standards und ihre Genauigkeitsanforderungen (Gröger, Kolbe, Nagel & Häfele, 2012; Albert, Bachmann & Hellmeier, 2003). 83

Tabelle 8 Abbildungstabelle der Eingangsdaten auf Module von CityGML2.0 und CityGML3.0. 96

Tabelle 9 Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” thematischer Gebäudeflächen. 113

Tabelle 10 Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” für Gebäudeinstallationen. 113

Tabelle 11 Zulässige Code-Listen-Werte der Attribute ”class”, ”function” für Gebäude/-Gebäudeteile. 114

Tabelle 12 Code-Listen-Werte des Attributs ”relationType” für Objekt-Relationen. 114

Tabelle 13 CityGML-Zielklassen der im Lageplan enthaltenen Objektarten. 144

Tabelle 14 Resultierende Shape-Dateien nach der Verarbeitung des CAD (computer-aided design)-Lageplans in QGIS. 146

Tabelle 15 Eingangsdaten mit Format und Herkunft. 146

Inhaltsverzeichnis

Abstract i
Zusammenfassung ii
1 Einführung 1
1.1 Motivation 1
1.1.1 Digitalisierung und Automatisierung im öffentlichen Sektor 1
1.1.2 Digitalisierung und Automatisierung in der Bauplanung und im Baugenehmigungsverfahren 3
1.2 Forschungsfragen 4
1.2.1 Datenmodell eines Lageplans in 3D 4
1.2.2 Automatisierte Abstandsflächenberechnung 6
1.3 Ziele der Arbeit 7
1.3.1 SiteplanADE 7
1.3.2 Automatisierte Abstandsflächenberechnung 7
1.3.3 Workflow 7
2 Die Rolle des Lageplans im Baugenehmigungsverfahren 8
2.1 Das derzeitige Baugenehmigungsverfahren in Deutschland 8
2.1.1 Der Ablauf im Baugenehmigungsverfahren 8
2.1.2 Die Rolle des Lageplans im Baugenehmigungsverfahren 10
2.2 Entwicklungen in der Digitalisierung und Automatisierung im Baugenehmigungsverfahren 10
2.2.1 Entwicklungen in Europa 10
2.2.2 Entwicklungen in Deutschland 14
2.2.3 Zusammenfassung der generellen Idee des digitalisierten und automatisierten Baugenehmigungsverfahren 16
3 Das Datenmodell des digitalen 3D-Lageplans 17
3.1 Vorarbeit 17
3.1.1 Eingangsdaten für die Erstellung eines (amtlichen) Lageplans 17
3.1.2 Wahl eines Datenmodells als Basis für die Entwicklung des Datenmodells eines 3D-Lageplans 26
3.1.3 Anforderungen an das 3D-Lageplan-Datenmodell 34
3.1.4 Thematische Aufteilung der Lageplan-Objekte 52
3.1.5 Geometrie der Lageplan-Objekte in verschiedenen Lageplan-Versionen 57
3.2 Entwicklung des 3D-Lageplan-Modells: SiteplanADE 58
3.2.1 Generelles Konzept für die Modellierung der Lageplan-Objekte 58
3.2.2 Package: Core 60
3.2.3 Package: Building 60
3.2.4 Package: RiskAssessment 62
3.2.5 Package: RegulationConflict 64
3.2.6 Package: LandUsePlanning 66

3.2.7 Package: LandAdministrationCadastre 70
3.2.8 Package: ConstructionLaw 73
3.2.9 Package: ThematicGrouping 74
4 Automatisierte Abstandsflächenberechnung 75
4.1 Vorarbeit 75
4.1.1 Abstandsflächen im öffentlichen Baurecht 75
4.1.2 Eingangsdaten für die automatisierte Berechnung von Abstandsflächen 78
4.2 Implementierung der automatisierten Abstandsflächenberechnung 99
4.2.1 Einführung zum Algorithmus der automatisierten Abstandsflächenberechnung 99
4.2.2 CityGML-Gebäude-Profile für die automatisierte Abstandsflächenberechnung 101
4.2.3 Programm-internes Datenmodell 118
4.2.4 Prozess - Vorverarbeitung 122
4.2.5 Prozess - Untersuchung auf Abstandsflächen 125
4.2.6 Unterprozess - Prüfung, ob Abstandsfläche erforderlich 126
4.2.7 Unterprozess - Berechnung der Abstandsflächen 132
4.2.8 Implementierung im Safe Software FME Transformer ”PythonCaller” 135
4.2.9 Anwendung des Algorithmus zur Abstandsflächenberechnung auf verschiedene Gebäudemodelle 139
5 Erstellung eines 3D-Lageplans 141
5.1 Workflow 141
5.2 Durchführung/Implementierung des Workflows 142
5.2.1 Vorverarbeitung des CAD-Lageplans 142
5.2.2 Prozessierung in der Feature Manipulation Engine 146
5.2.3 Resultat der Erstellung eines 3D-Lageplans 158
6 Diskussion 167
6.1 Diskussion der Forschungsfragen 167
6.1.1 Datenmodell eines Lageplans in 3D 167
6.1.2 Automatisierte Abstandsflächenberechnung 172
6.2 Diskussion der Ziele der Arbeit 176
6.2.1 SiteplanADE 176
6.2.2 Automatisierte Abstandsflächenberechnung 177
6.2.3 Workflow 178
7 Ausblick 180
7.1 Ausblick in Bezug auf den digitalen Lageplan in 3D 180
7.2 Ausblick in Bezug auf die automatisierte Abstandsflächenberechnung 182
7.3 Zusammenfassender Ausblick mit Bezug auf semantische Datenmodelle und Automatisierung 183
A Gebäude-Profil CityGML 2.0 184
B Gebäude-Profil CityGML 3.0 239
C SiteplanADE Objektartenkatalog 269
D Python Dokumentation zum Algorithmus der Abstandsflächenberechnung 594
E Anleitung zur Konvertierung eines CAD-Gebäudeentwurfs in ein CityGML-Modell in SketchUp 658

F Verzeichnis des digitalen Anhangs 674
Literatur 682

1 Einführung

1.1 Motivation

1.1.1 Digitalisierung und Automatisierung im öffentlichen Sektor

Mit zunehmender Digitalisierung von Dienstleistungen und gesellschaftlichen Abläufen ändern sich auch die Anforderungen und Erwartungen an den Service des öffentlichen Sektors. Öffentliche Einrichtungen und deren Dienstleistungen sollen zu jeder Zeit online zugänglich sein, außerdem eine schnelle Prozessierung der Anfrage bieten (Balka, Daub & Pflanzer, 2018). Die wachsenden Anforderungen stellen den öffentlichen Sektor, im Vordergrund des prognostizierten Fachkräftemangels, vor eine schwierige Aufgabe. Die Aufgabenbewältigung in öffentlichen Einrichtungen würde zunehmend ein unmögliches Unterfangen werden und ohne technologische Unterstützung in einem Service von schlechter Qualität und langen Bearbeitungszeiträumen münden (Balka et al., 2018).

Die Digitalisierung und Automatisierung bietet das Potenzial, den Sachbearbeiter*innen sich wiederholende und regelbasierte Aufgaben abzunehmen und diese von Maschinen zeiteffizienter und zugleich weniger fehleranfällig bearbeiten zu lassen (Balka et al., 2018). Das hätte den Vorteil, die Arbeitszeit der Angestellten auf die Betreuung von Bürger*innen und Kund*innen fokussieren zu können, was von der Gesellschaft positiv aufgenommen werden würde (Balka et al., 2018). Zugleich könnte der zukünftige Fachkräftemangel kompensiert werden, indem automatisierbare Aufgabenbereiche nun von Maschinen erledigt werden und somit weniger Personal benötigt wird (Balka et al., 2018). Laut einer Studie von McKinsey gäbe es demnach im öffentlichen Dienst durchschnittlich das Potenzial bis zu 42% der Tätigkeiten zu automatisieren (siehe Abbildung 1) (Balka et al., 2018). Neben sich wiederholenden körperlichen Tätigkeiten können v.a. (vor allem) in der Datenverarbeitung und -erfassung bis zu 59% der Aktivitäten automatisiert werden. In der öffentlichen Verwaltung bietet bis zu 64% der Arbeitszeit Automatisierungspotenzial (siehe Abbildung 2) (Balka et al., 2018).

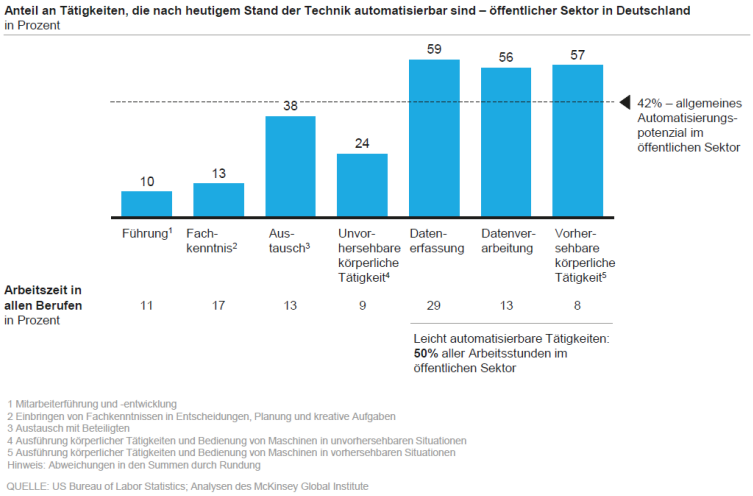


Abbildung 1 Tätigkeitsbereiche, die Potenzial für Automatisierung bieten.

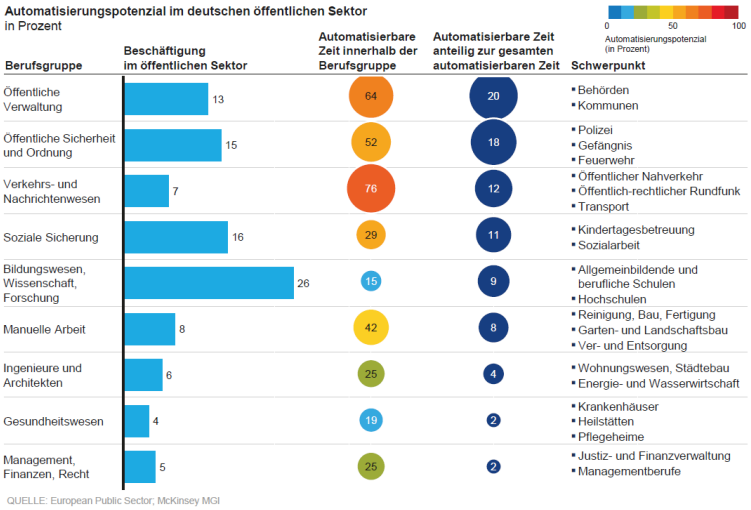


Abbildung 2 Automatisierbare Arbeitszeit nach Berufsgruppen im öffentlichen Dienst.

Das Potenzial der Digitalisierung und Automatisierung von Verwaltungsprozessen liegt des weiteren darin begründet, dass Verwaltungsleistungen auf wiederkehrenden Entscheidungen, welche eindeutigen Regelungen zugrunde liegen, basieren (Raffer, 2019). Verwaltungsprozesse und Entscheidungen lassen sich über BPMN (Business Process Model and Notation) bzw. DMN (Decision Model and Notation) standardisiert Beschreiben und können von Regelmaschinen prozessiert werden (Kühn, Plazek, Schuster, Czanderle & Peper, 2018). Die Standardisierung der Beschreibung und Durchführung der Verwaltungsprozesse hat den positiven Effekt, dass die Verwaltungsentscheidungen durch die maschinelle Bearbeitung schnell bereitstehen, ihnen außerdem einheitlicher Charakter und hohe Qualität verliehen werden (Kühn et al., 2018).

Eine (Teil-)Automatisierung der Arbeitsschritte kann den gesamten Arbeitsablauf von der Prüfung der Eingangsdaten, über die Entscheidung selbst, bis hin zur Bekanntmachung der Entscheidung betreffen (Kühn et al., 2018).

1.1.2 Digitalisierung und Automatisierung in der Bauplanung und im Baugenehmigungsverfahren

Steht ein Bauvorhaben an, so ist nach Bauvorlagenverordnung der Länder unter anderem die Erstellung eins Lageplans im Kontext der baulichen Anlage und ihrer näheren Umgebung erforderlich. Dieser soll neben dem/n geplanten Gebäudeobjekt/en, Objekte aus Baurecht, Liegenschaftskataster und Bebauungsplan, sowie andere topographische und für das Bauvorhaben relevante Objekte wie z.B. Vegetationsbestände und Versorgungsleitungen beinhalten. Noch wird der Lageplan als zwei-dimensionaler Lageplan in CAD-Formaten bei der Prüfstelle eingereicht, obwohl in der Architektur und im Bauwesen das Planen und Entwerfen von Bauwerken im Zuge von BIM immer weiter in Richtung der 3D-Modellierung geht (Jones & Laquidara-Carr, 2018; Ding, Zhou & Akinci, 2014; Becker, Clemen & Wunderlich, 2019). Folglich müssen auch die Vermessungsingenieur*innen vermehrt die von den Architekt*innen und Bauingenieur*innen geforderten Daten in entsprechenden 3D-Modellen bereitstellen (Becker et al., 2019). Aufgrund dieser Entwicklung wäre es sinnvoll, auch die Lagepläne zum Baugesuch in dreidimensionalen Datenmodellen anzufertigen. Der Vorteil einer Überführung eines Lageplans in ein 3D-Datenmodell ist, dass dieser in einer Software visualisiert und begutachtet werden kann, sowie Architekt*innen/Bauingenieur*innen als Grundlage des Bauwerkdesigns dienen kann (Noardo et al., 2019a).

Auch für die Digitalisierung und Automatisierung hinsichtlich der Prüfung und Genehmigung von Bauvorhaben sind semantische 3D-Modelle des Lageplans notwendig. Semantische 3D-Modelle des Lageplans können von entsprechenden Algorithmen interpretiert werden und wären als Eingangsdaten in eine digitalisierte (semi-)automatische Baugenehmigungsprüfung in Aussicht zu stellen (Becker et al., 2019).
Zudem machen verschiedene Ansichten und Perspektiven, die durch das 3D-Modell mit geeigneter Software möglich werden, Berechnungen des*der Vermessungsingenieur*innen und enthaltene Geometrien leichter nachvollziehbar.
Der 3D-Lageplan bietet außerdem das Potenzial, die Akzeptanz und Nachvollziehbarkeit für Bauprojekte in der Bevölkerung durch Bereitstellung von interaktiven Visualisierungen über das Internet erhöhen (Egger, 2019).

Der BDVI möchte bei der die Digitalisierung und Automatisierung der Arbeitsschritte schon bei der Erstellung eines baurechtlichen Lageplans ansetzen. Als eine wichtige Komponente im Automatisierungsprozess wird eine automatische Berechnung der Abstandsflächen des Baukörpers anhand eines Gebäudemodells gesehen, der im Vorgang der Produktion von 3D-Lageplan-Modellen eingebunden ist.

1.2 Forschungsfragen

1.2.1 Datenmodell eines Lageplans in 3D

Im Datenmodell zu modellierende Objekte

Um ein Datenmodell zu entwerfen, das möglichst alle relevanten Objekte eines Lageplans und deren Eigenschaften abdeckt, ist es notwendig, die aktuelle Handhabung bei der Erstellung von Lageplänen im Zuge des Baugenehmigungsverfahren, sowie zukünftige Entwicklungen (z.B. Digitalisierung/Automatisierung im Baugenehmigungsverfahren) bei dem Entwurf des Datenmodells zu berücksichtigen und entsprechend zu implementieren. Dafür sind die erforderlichen Eingangsdaten zu definieren. Außerdem ist zu hinterfragen, ob die aktuell zu erfassende Objektmenge, welche der Lageplan beinhaltet statisch ist, oder durch Entwicklungen in der Stadtplanung und Innovationen im Stadtbild dynamischen Änderungen unterworfen ist.

Thematische Aufteilung der Lageplan-Objekte

Es stehen mehrere thematische Gliederungen der Lageplan-Objekte zur Diskussion, welche mit Bezug zur Anwendung in der Praxis zu bewerten sind. Es besteht die Möglichkeit, die im Lageplan enthaltenen Objekte nach bestehenden und geplanten Objekten in jeweils Topographie-Objekte und virtuell-konzeptionelle (Land-administrative/privat-rechtliche/öffentlich-rechtliche) Objekte aufzuteilen. Ein anderer Ansatz geht von der Datenherkunft aus und teilt die Objekte im Lageplan deren Zugehörigkeiten (z.B. Objekte des Bebauungsplans, Objekte des Katasterplans) zu. Die Bewertung der Notwendigkeit und der sich ergebenden Anwendungsfälle, die die beiden Aufteilungsansätze mit sich bringen, sollen darüber entscheiden, welcher Ansatz schlussendlich im Datenmodell verfolgt werden soll.

Wahl der Geometrie der Lageplan-Objekte

Es ist zu untersuchen, in welchem LOD und als welcher Geometrietyp die Lageplan-Objekte (planerische Festsetzungen/rechtliche Objekte, sowie topographische Objekte) dargestellt werden sollen.

Verschiedene Varianten des Lageplans

Aufgrund verschiedener Anforderungen an Lagepläne (z.B. beurkundete und damit amtliche Lagepläne) kann sich der Anspruch an die Auswahl von dargestellten Objektarten unterscheiden. Laut der BauPrüfVO (Verordnung über bautechnische Prüfungen) NRW (Nordrhein-Westfalen) wird eine Unterscheidung zwischen dem amtlichen und dem nicht amtlichen Lageplan getroffen.

Nach BauPrüfVO NRW §3 Absatz 3 mit Stand vom 24.7.2020 muss ein amtlicher Lageplan von einem Katasteramt oder einem/er ÖBVI (Öffentlich bestellter Vermessungsingenieur) erstellt werden, wenn ein amtlicher Lageplan angefordert wurde, oder die Voraussetzungen aus BauPrüfVO §3 Absatz 3 Satz 1 gegeben sind. Insbesondere wenn,

> *„[...]*
> *1. es sich bei den Außengrenzen des Baugrundstücks nicht um festgestellte Grenzen im Sinne des Vermessungs- und Katastergesetzes vom 1. März 2005 (GV. NRW. S. 174) in der jeweils geltenden Fassung handelt,*
> *2. die Grenzen des Baugrundstücks und die vorhandenen baulichen Anlagen auf dem Baugrundstück und den angrenzenden Grundstücken so vermessen sind, dass für die Grenzpunkte Koordinaten in einem einheitlichen System nicht ermittelt werden können, oder*
> *3. auf dem Baugrundstück oder von angrenzenden Grundstücken her Grenzüberbauungen vorliegen,*
> *4. eine Baulast im Sinne von § 18 auf dem Baugrundstück oder eine das Baugrundstück betreffende Baulast auf den angrenzenden Grundstücken ruht. Dies führt zur Fragestellung, wie diese Auswahl der Objektarten, je nach Anforderung und Variante des Lageplans, im Datenmodell berücksichtigt werden kann.*
> *[...]"*
> (BauPrüfVO NRW §3 Absatz 3 mit Stand vom 24.7.2020)

D.h. (das heißt), sobald neue Einmessungen für Grenzen, Grenzpunkte, bauliche Anlagen, Grenzüberbauungen notwendig werden oder Baulasten auf den betreffenden Grundstücken liegen, muss dies in einem amtlichen, also einem *„[...] mit öffentlichem Glauben beurkundet[n] [...]"* (BauPrüfVO NRW §3 Absatz 3 Satz 1 mit Stand vom 24.7.2020) Lageplan aufgenommen und dargestellt werden.
Die vorhergehende Darstellung führt zur Fragestellung, wie diese Auswahl der Objektarten, je nach Anforderung und Variante des Lageplans, im Datenmodell berücksichtigt werden kann, und, ob dies erforderlich ist.

Ableitung eines 2D-Lageplans vom 3D-Lageplan

Auf Grund der momentanen Praxis in der Baugenehmigungsprüfung ist es notwendig, eine Transformation des 3D-Lageplans in einen 2D-Plan zu gewährleisten (Becker et al., 2019). Es muss deshalb evaluiert werden, wie weit das Datenmodell des Lageplans diesen Bearbeitungsschritt unterstützen soll. Insbesondere, ob neben der 3D-Repräsentation des Lageplans auch eine 2D-Repräsentation im Datenmodell vorhanden sein soll. Oder, wie eine automatisierter Ableitung des 2D-Lageplans aus einem 3D-Lageplan realisiert werden könnte.

Elektronische Signatur des XML-Dokuments

Um das XML-Dokument eines Lageplans elektronisch beurkunden zu können, muss es möglich sein der Datei eine qualifizierte elektronische Signatur beizufügen. Für XML-Dokumente entwickelte eine Arbeitsgruppe des W3C (World Wide Web Consortium) hierfür das Signatur-Format XMLDSig (XML Digital Signature) (Hühnlein & Korte, 2006; Yiu et al., 2015; Roessler et al., 2013). Mit XAdES (XML Advanced Electronic Signatures) wurde eine Erweiterung der XMLDSig-Spezifikation mit Konformität zu den europäischen Richtlinien zu elektronischen Signaturen geschaffen (Cruellas, Karlinger, Pinkas & Ross, 2003; *TS 101 903 - V1.3.2 - XML Advanced Electronic Signatures (XAdES)*, 2006). Es muss betrachtet werden, inwieweit diese Technologie für den Anwendungsfall des Lageplan-Dokuments zum tragen kommt.

1.2.2 Automatisierte Abstandsflächenberechnung

Erforderliche Eingangsdaten

Zu Beginn ist es notwendig, zu definieren, welche Eingangsdaten neben dem Gebäudemodell für die Berechnung von Abstandsflächen notwendig sind. Auch, in welchem Datenformat diese Daten für die Berechnung vorliegen ist von Bedeutung und muss bewertet werden.

Level of Detail des Gebäudemodells für die Abstandsflächenberechnung

Im Zuge der Arbeit gilt es zu klären, welches geometrische LOD das Gebäudemodell für die Abstandsflächenberechnung aufweisen muss, da auch Gebäudeinstallationen, Vorbauten und Gesimse bzgl. (bezüglich) der Abstandsflächenberechnung relevant sind und deshalb im Gebäudemodell enthalten sein müssen. Gebäude-interne Objekte wiederum können vernachlässigt werden, da nur die äußere Hülle des Gebäudes für die Berechnung von Interesse ist.

Geometrische Repräsentation des Gebäudemodells für die Abstandsflächenberechnung

Für die Bereitstellung von Gebäudemodellen existieren verschiedene geometrische Repräsentationen abhängig davon, aus welchem Fachbereich das Modell stammt. Aus der Architektur und dem Bauwesen werden Gebäude mit Volumenkörpern modelliert, in der Geoinformation ist die Modellierung über Begrenzungsflächen üblich (Kaden et al., 2019). Welche Repräsentation für die Berechnung der Abstandsflächen verwendet werden soll, muss untersucht werden.

1.3 Ziele der Arbeit

1.3.1 SiteplanADE

Die Datenmodellerweiterung des CityGML3.0-Modells um den Anwendungsfall ”Amtlicher Lageplan” (SiteplanADE) soll es ermöglichen, einen offiziellen amtlichen Lageplan zu instanzieren. Nationale Standards aus dem öffentlichen Bereich wie AAA und XPlanung werden deshalb bei der Entwicklung des Datenmodells nachempfunden. Trotzdem soll die internationale Reichweite des CityGML Standards berücksichtigt und Interoperabilität zu europäisch bzw. international existierenden Standards wie INSPIRE bzw. LADM (Land Administration Domain Model) gewährleistet werden, sodass die ADE (Application Domain Extension) möglicherweise auch international anwendbar ist.

1.3.2 Automatisierte Abstandsflächenberechnung

Die Abstandsflächenberechnung soll am Gebäudemodell im CityGML-Datenformat erfolgen und aus DGM, Grundstücksgrenzen, Baugebietsflächen, sowie rechtlicher Grundlage, Abstandsflächen berechnen und auf entsprechende Klassen der SiteplanADE abbilden. Die Abstandsflächen sollen als 3D-Polygone den zugehörigen Gebäudekomponenten zugeordnet werden.

1.3.3 Workflow

Ein beispielhafter Workflow zu Erstellung eines 3D-Lageplans unter Berücksichtigung verschiedener Eingangsdaten (Architektenpläne, DGM, Liegenschaftskarte, Bebauungsplan, etc. (et cetera)) in verschiedenen Datenformaten soll bereitgestellt werden. Die Realisierung des Workflows sollen so weit möglich in der FME von Safe Software geschehen.

2 Die Rolle des Lageplans im Baugenehmigungsverfahren

2.1 Das derzeitige Baugenehmigungsverfahren in Deutschland

2.1.1 Der Ablauf im Baugenehmigungsverfahren

Der gegenwärtige Ablauf des Baugenehmigungsverfahrens wird in Abbildung 3 dargestellt. Vorbereitend werden auf Basis der geltenden Bauvorlagenverordnung alle notwendigen Dokumente zusammengetragen. Diese beinhalten u.a. (unter anderem) Bestandspläne, Lagepläne, Bauzeichnungen, Baubeschreibungen, Standsicherheitsnachweise und Brandschutznachweise. Die Unterlagen können je nach Baubehörde in Papierform postalisch, persönlich oder elektronisch eingereicht werden. In der Baubehörde eingegangen, werden die Antragsdokumente auf Vollständigkeit und Richtigkeit überprüft, ggf. (gegebenenfalls) werden Nachbesserungen seitens des*der Antragsteller*in fällig. Sind die Dokumente vorerst vollständig und entsprechen den Prüfkriterien, werden weitere Beteiligte in die Bauantragsprüfung miteinbezogen. Dies sind Behörden und Gremien, wie z.B. Natur- und Umweltschutzbehörden oder Denkmalschutzbehörden. Sie fertigen Stellungnahmen an, welche bei der weiteren Prüfung und Abstimmung der technischen Entscheidung über die Erteilung einer Baugenehmigung berücksichtigt werden. Negative Stellungnahmen können wiederholt zur Notwendigkeit von Änderungen im Bauantrag führen. Erlauben die Dokumente und Stellungnahmen schlussendlich eine positive Entscheidung, wird von der Baubehörde eine Baugenehmigung ausgestellt.

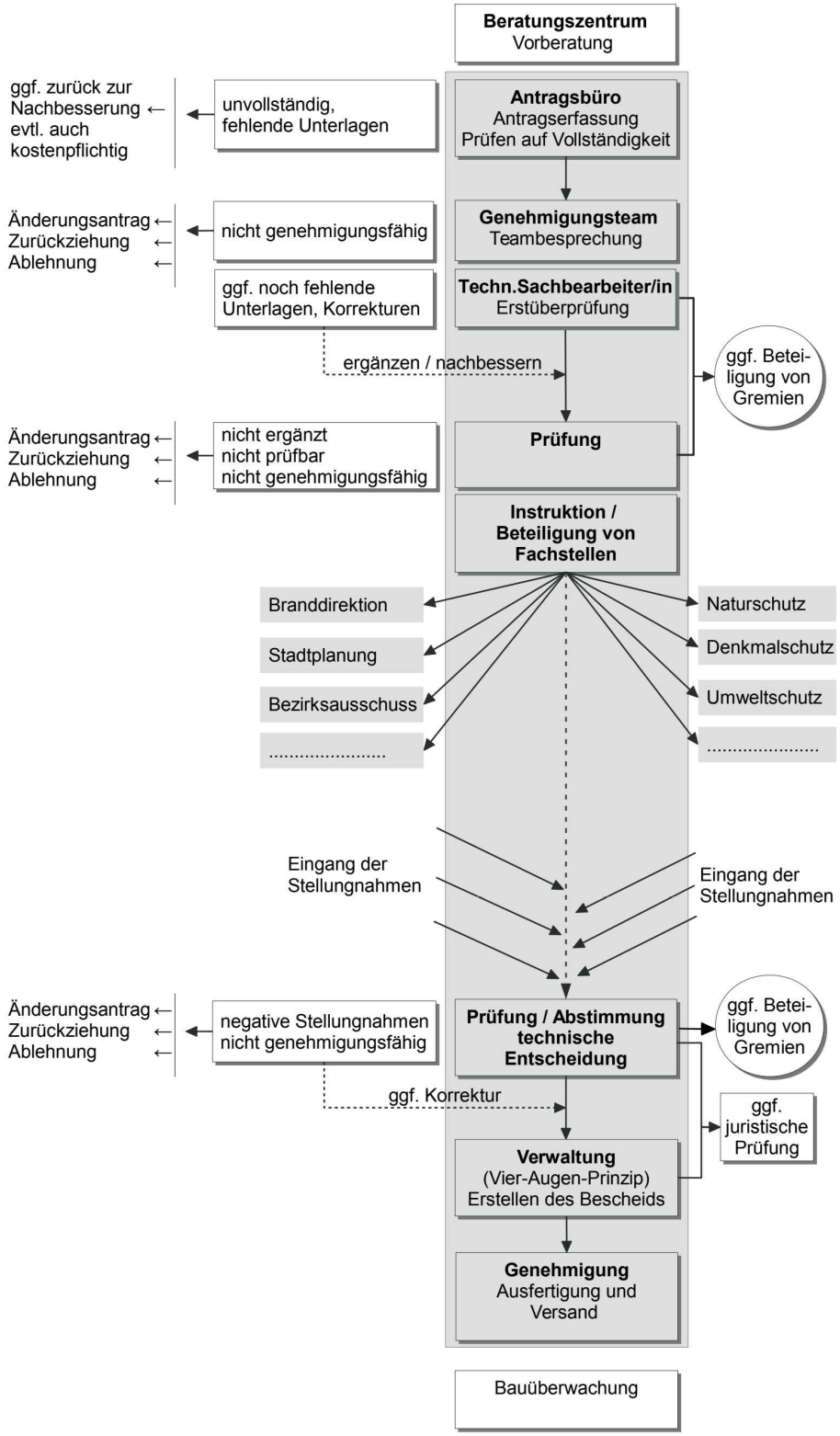


Abbildung 3 Schema des Ablaufs im Baugenehmigungsverfahren (Landeshauptstadt München Referat für Statdplanung und Bauordnung Lokalbaukomission, 2017).

2.1.2 Die Rolle des Lageplans im Baugenehmigungsverfahren

Ziel des (amtlichen) Lageplans ist es, der Baubehörde einen Gesamtüberblick über das geplante Bauvorhaben zu schaffen. Der Lageplan macht es möglich, eine grundsätzliche Einschätzung und Beurteilung über das Bauvorhaben abzugeben (Noack et al., 2005). Zudem dient dieser als Grundlage zur Überprüfung der Einhaltung von Festsetzungen, die das Baurecht oder öffentliche Regulierungen betreffen, sowie der Sicherstellung von öffentlicher Ordnung und Sicherheit (Noack et al., 2005). Inhalt des Lageplans ist deshalb neben vorhandenen und geplanten topographischen Objekten (z.B. vorhandene bauliche Anlagen, Baumbestand, etc.) konzeptionelle Konstrukte (z.B. Baulasten, Abstandsflächen, Baulinien, Baugrenzen, Art und Maß der baulichen Nutzung, etc.) bzgl. des zu bebauenden Grundstücks (Noack et al., 2005). Auch wegen seiner hohen Genauigkeit dient der Lageplan schon dem*der Entwurfsverfasser*in als Planungsgrundlage, um das Bauobjekt entsprechend der örtlichen Begebenheiten zu entwerfen (Noack et al., 2005).
Der amtliche Lageplan, erstellt von einem ÖBVI, hat den Charakter einer öffentlichen Urkunde und schafft damit Rechtssicherheit im gesamten Verlauf des Bauvorhabens (Noack et al., 2005).

2.2 Entwicklungen in der Digitalisierung und Automatisierung im Baugenehmigungsverfahren

2.2.1 Entwicklungen in Europa

EuroSDR GeoBIM Project

Das GeoBIM Projekt der Technischen Universität Delft in Kooperation mit der nicht-kommerziellen Forschungsinstitution EuroSDR treibt die Entwicklung eines internationalen Standards bei der Integration von GeoBIM in administrativen Bereichen voran (Noardo et al., 2019a). Dabei wurde u.a. ein Workflow entwickelt, der die Integration von BIM und Geoinformation in den verschiedenen Phasen eines Baugenehmigungsverfahrens abbildet (siehe Abbildung 4).
Für die vorbereitende Beratung des Bauvorhabens würden dabei existierende digitale 3D-Stadtmodelle und rechtliche Regulierungen in geeigneten Softwares zur Visualisierung und groben Planung zusammengeführt werden (Noardo et al., 2019a).
Das sich anschließende Baudesign würde mittels BIM durchgeführt werden, wobei für die Beachtung der gesetzlichen Vorgaben digital bereitgestellte Stadtmodelle und Regulierungen als Basis dienen, und in die entsprechende Editoren-Software hinzugeladen werden könnten (Noardo et al., 2019a). Die Antragsabgabe des Gebäudeentwurfs würde dann im BIM-Datenmodell erfolgen (Noardo et al., 2019a).
Für den Einbezug der Nachbarn in das Bauprojekt würde das BIM-Planungsmodell provisorisch in das bestehende 3D-Stadtmodell integriert werden (Noardo et al., 2019a). Die Veröffentlichung des Bauvorhabens, sowie des Projektfortschritts, erfolgt ebenfalls durch das BIM-Modell (Noardo et al., 2019a). Der LOD* (Level of Development)-Parameter dient der Kenntlichmachung des Entwicklungsstands des Bauprojektes (Noardo et al., 2019a).

Bei der formellen Prüfung des Bauantrags würde eine Validierung des BIM-Gebäudemodell vor dem Hintergrund festgelegter Richtlinien für die Modellierung stattfinden (Noardo et al., 2019a). D.h., das digitale Gebäudemodell würde auf zulässige Geometrien, Semantik und Georeferenzierung geprüft werden (Noardo et al., 2019a). Die materielle Prüfung des Bauantrags und technische Stellungnahmen zum Bauvorhaben würden nach positivem Feedback über die formelle Richtigkeit des Antrags eintreten. Analysen bzgl. Brandschutz, Statik u.ä. (und Ähnliches) würden direkt am BIM-Gebäudemodell durchgeführt werden (Noardo et al., 2019a).

Für die Validierung des geplanten Gebäudes auf Einhaltung der öffentlich-rechtlichen Festsetzungen, sowie privat-rechtlicher Begebenheiten, würde das BIM-Gebäudemodell in das existierende 3D-Stadtmodell eingebettet und mit digitalisierten Regulierungen überlagert werden (Noardo et al., 2019a).

Während der Bauphase kann am BIM-Modell der aktuelle Baufortschritt dokumentiert und fortlaufend auf baurechtliche Verstöße untersucht werden (Noardo et al., 2019a). Die endgültige Integration in das 3D-Stadtmodell würde nach Abschluss des Baus erfolgen, nachdem der finale Zustand des Bauwerks nochmals als BIM-Gebäudemodell erfasst worden ist (Noardo et al., 2019a).

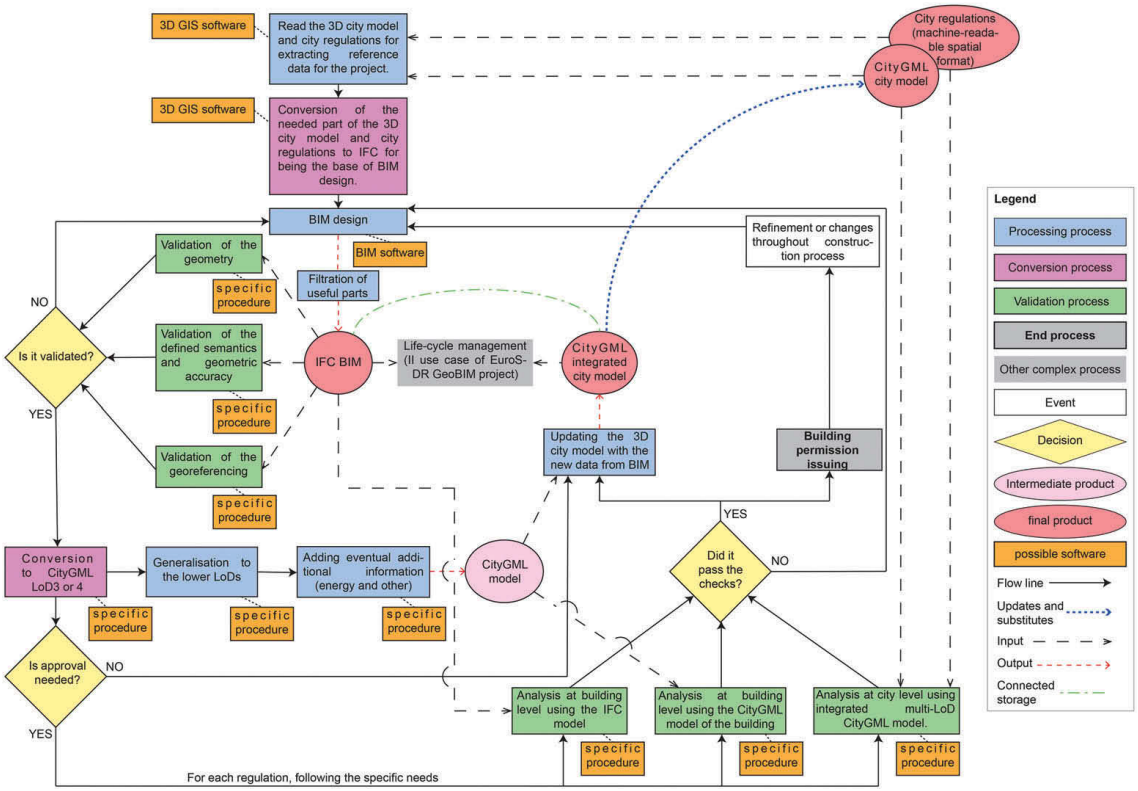


Abbildung 4 Workflow für das Baugenehmigungsverfahren unter Anwendung des GeoBIM-Ansatzes (Noardo et al., 2019a).

Die Rolle des Lageplans im EuroSDR GeoBIM Project

Wie in Abbildung 6 zu sehen, ist die Anwendung des digitalen, drei-dimensionalen Lageplans in planerischen, sowie Bauvorlagen-prüfenden Arbeitsschritten zu finden. Auf planerischer Stufe würde der 3D-Lageplan dem*der Entwurfsverfasser*in als Basis bei der Planung und des Designs des Bauwerks dienen (siehe Abbildung 5). Damit können schon in frühen Phasen der Planung Fehler detektiert und damit die Kosten und Verzögerungen im Bauprozess minimiert werden (Egger, 2019).

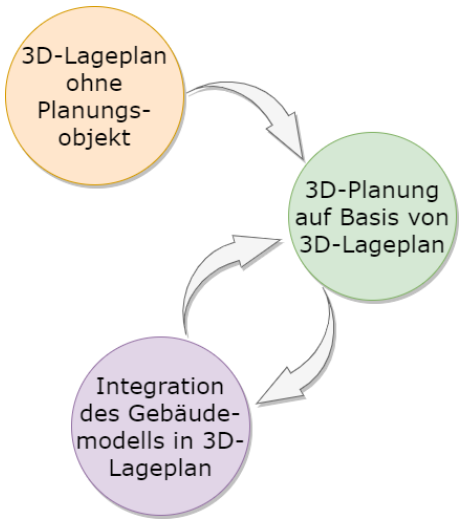


Abbildung 5 Anwendungsfall des Lageplans im Prozess der Bauwerksplanung auf Basis des Lageplans.

Die automatisierte Berechnung der Abstandsflächen und die Anfertigung des Lageplans würde hingegen Bestandteil eines Bearbeitungsschritts vor der Aktivität "Baugenehmigung" (*Building permission issuing*) des abgebildeten Arbeitsablaufs in Abbildung 6 sein. Der Lageplan mit den erfassten rechtlichen Objekten ginge sowohl in den Prozess der Bauwerksanalyse auf Bauwerksebene, als auch in die Analyse auf Stadtebene, ein (siehe Abbildung 4).

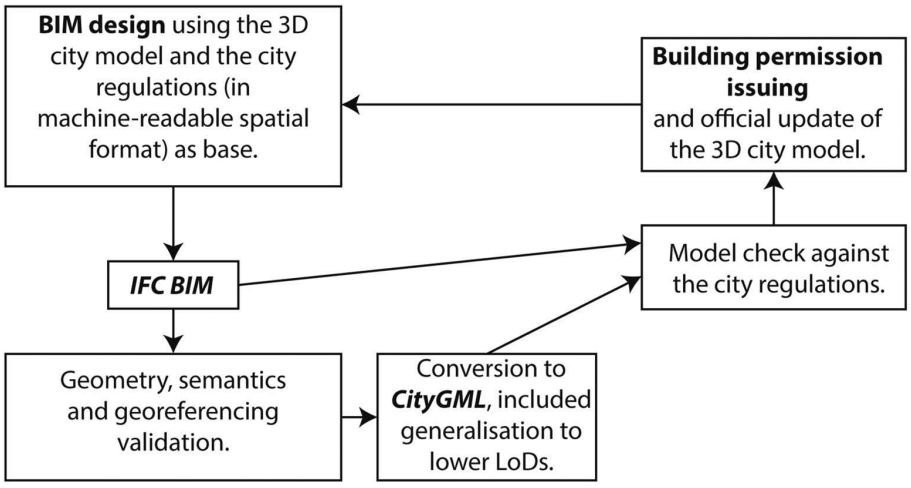


Abbildung 6 Vereinfachter Workflow für die Bauplanung mit GeoBIM (Noardo et al., 2019a).

Europäisches Netzwerk für digitale Baugenehmigungsverfahren (EUnet4DBP)

Das EUnet4DBP (European Network for Digital Building Permits) ist ein Zusammenschluss verschiedener Institutionen aus dem öffentlichen, sowie privat-wirtschaftlichen Bereich des AEC (Architecture, Engineering and Construction)-Sektors. Ziel dieses Projekts ist, eine gemeinsame Entwicklung von Strategien für die Digitalisierung des Baugenehmigungsverfahrens voranzutreiben (Noardo et al., 2020). Im Fokus steht dabei die gemeinsame Definition von Software-Tools und Methoden in einem interoperablen Workflow des Baugenehmigungsverfahrens (Noardo et al., 2020). Aufgrund von Erfahrungsberichten von Expert*innen wurden drei Säulen für das EUnet4DBP-Projekt festgelegt, welche sich auf den *Prozess*, die *Regeln und Anforderungen* und die *Technologie* im Baugenehmigungsverfahren erstrecken (Noardo et al., 2020).
Die Säule des *Prozesses* bezieht sich auf ausgeführte Praktiken und bürokratischen Arbeitsketten, welche an die Digitalisierung angepasst werden müssen (Noardo et al., 2020). *Regeln und Anforderungen* bezeichnet Kriterien und Richtlinien, die befolgt werden sollen, um die Ziele des Anwendungsfalls *Baugenehmigungsverfahren* zu erreichen (Noardo et al., 2020). Letztlich sollen in der *Technologie*-Säule die vorigen Säulen implementiert werden (Noardo et al., 2020).
Im Zuge eines Workshops erarbeiten die Mitglieder des EUnet4DBP einige Prinzipien für die drei Säulen bei der Digitalisierung des Baugenehmigungsverfahrens:

- Effizienz im Baugenehmigungsverfahren
- Interoperabilität von angewandten und übernommenen Technologien innerhalb eines *Open Data Frameworks*
- Maschinen-lesbarer Ansatz, um eine größtmögliche Automatisierung im Prozess zu realisieren

(Noardo et al., 2020)

In den daraus folgenden formulierten Zielen ergaben sich u.a. die Forderungen nach europäischer Angleichung im Baugenehmigungsverfahren, Verwendung von offenen Standards bei den Datenformaten, sowie einen automatisierten und Maschinen-lesbaren Ablauf (Noardo et al., 2020).

2.2.2 Entwicklungen in Deutschland

Projekt BIM-basierter Bauantrag

Im Forschungsprojekt *BIM-basierter Bauantrag* (http://www.bimbauantrag.de/) wird die Verknüpfung offener Standards wie XBau und XPlanung mit BIM-Modellen für einen digitale Bauantrag mit teilautomatisierter Prüfung untersucht (Theiler, Tulke, König & Krause, 2019). In Abbildung 7 ist dargestellt, wie die Beteiligten eines Bauantrages untereinander interagieren. Die öffentlichen Beteiligten wie Katasteramt/Landesvermessungsamt und Bauportal wären in ersten Schritten dafür zuständig, die Basisdaten für die Planung des Bauvorhabens bereitzustellen.
In Zusammenarbeit von Antragssteller*innen mit Fachplaner*innen und öffentlich bestellten Vermessungsingenieur*innen würden alle Unterlagen vorbereitet werden, die für den Antrag notwendig sind. Nach Erstellung des Antrags würde die automatisierte regelbasierte formelle und materielle Prüfung der Unterlagen stattfinden. Bei Fehlermeldungen hätte der*die Antragsteller*in seine Bauvorlagen zu überprüfen und nachzubessern. Würden bei der automatisierten Prüfung keine Warnungen oder Fehlermeldungen ausgegeben werden, so könnte der Antrag abgegeben werden.
Als nächster Schritt würde der Antrag manuell von einem*r Bauprüfer*in auf formelle/materielle Korrektheit und Vollständigkeit geprüft werden. Wieder könnte es zur Nachbesserung und zum Nachreichen von Dokumenten durch den*die Antragsteller*in kommen. Bestünde der Antrag die Prüfung mit einem positiven Ergebnis, so würde ein Bescheid erstellt werden, der festlegt, ob eine Baugenehmigung erteilt wird oder nicht.

Im Zuge dieses Forschungsprojekts wurde in einer prototypischen Software die Integration der Standards XPlanung und XBau mit BIM-Modellen im IFC (Industry Foundation Classes)-Datenformat implementiert (Theiler, 2020b). Die Software erlaubt es, die Daten zu editieren und als digitalen Bauantrag zu exportieren (Theiler, 2020b). Auch die Modellprüfung kann in dieser Software durchgeführt werden (Theiler, 2020b). Teil des Projektes war zusätzlich, die experimentelle Implementierung eines Webportals für das Einreichen des Bauantrags über das Internet (Theiler, 2020b). Das Webportal soll es ermöglichen, Feedback über Mängel, oder Anmerkungen über das Fehlen von Dokumenten an den*die Antragssteller*in zu übermitteln.

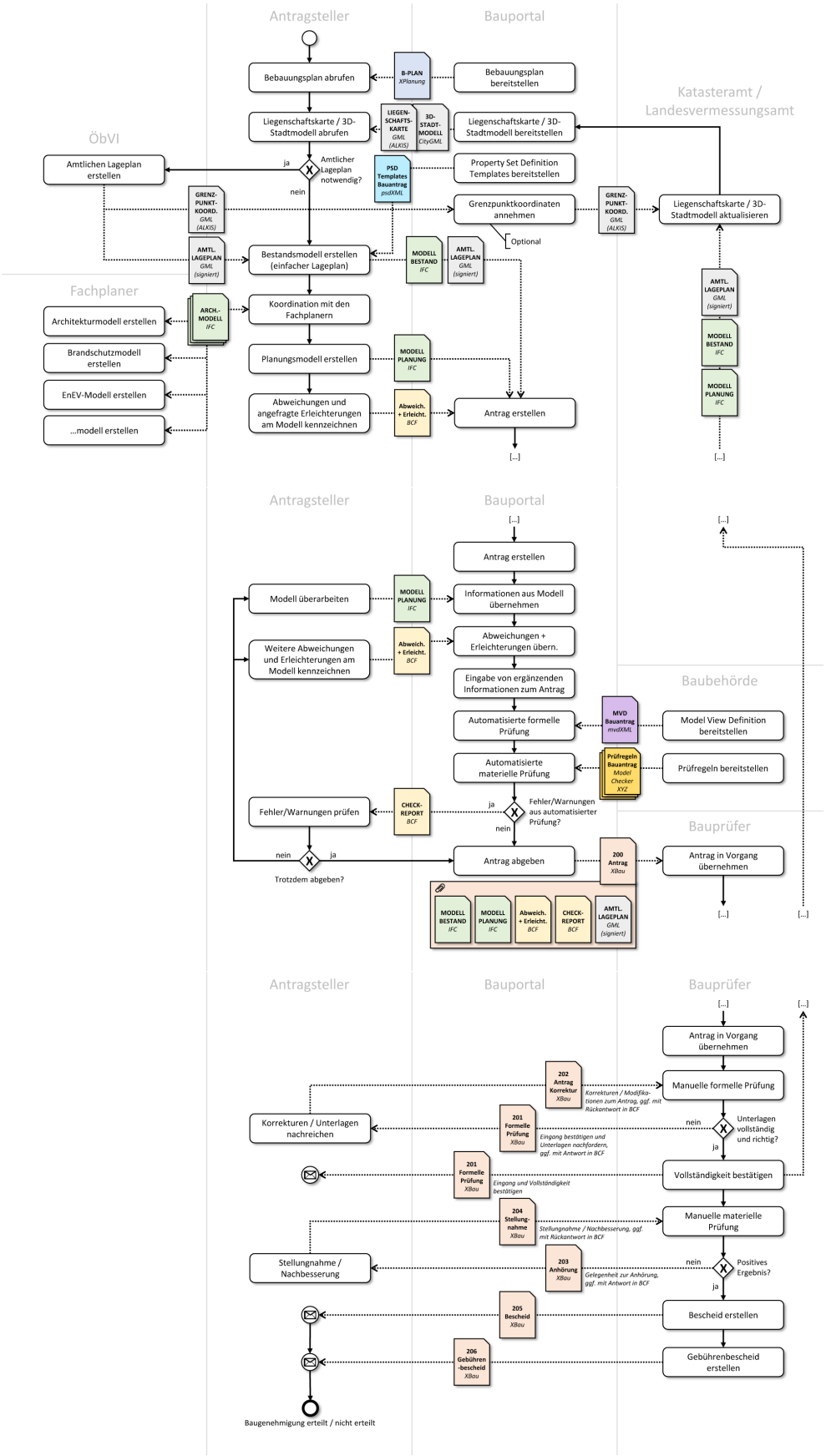


Abbildung 7 Workflow des BIM-basierten Bauantrags (Theiler, 2020a).

Die Rolle des Lageplans im BIM-basierten Bauantrag

Der in Abbildung 7 dargestellte Ausschnitt des Gesamtprozesses des BIM-basierten Bauantrags zeigt die Anwendungsfelder des digitalen Lageplans im Bereich des*der Antragsteller*in, des Bauportals, sowie den*der Fachplaner*innen.
Je nachdem, ob ein amtlicher Lageplan notwendig wäre oder nicht (abhängig von geltendem Recht [notwendig z.B. bei nicht festgestellte Grenzen oder vorhandenen Baulasten]) und des*der Sachbearbeiter*in), würde aufgrund eines einfachen oder amtlichen Lageplans ein Bestandsmodell erstellt werden. Dieses Bestandsmodell würde als Grundlage für die Entwurfsverfasser*innen dienen, um das Planungsmodell anzufertigen.
Nach dem Planungsprozess würde die geplante bauliche Anlage aus dem Planungsmodell in den Lageplan integriert und um erforderliche Berechnungen bzgl. der geplanten baulichen Anlage, wie z.B. Grundflächenzahl, Geschossflächenzahl, Zahl der Vollgeschosse, Baumassenzahl und Abstandsflächen ergänzt werden.
Des weiteren ist der (einfache/amtliche) Lageplan ein Bestandteil der notwendigen Bauvorlagen beim BIM-basierten Bauantrag, um das Bauvorhaben bzgl. der Festsetzungen auf städte-baulicher Ebene, sowie baurechtlichen und administrativen Gegebenheiten überprüfen zu können (Noack et al., 2005).

2.2.3 Zusammenfassung der generellen Idee des digitalisierten und automatisierten Baugenehmigungsverfahren

Die vorgestellten Projekte basiere auf der gemeinsamen Idee, das Baugenehmigungsverfahren und dessen Dokumente weitestgehend zu digitalisieren, um Potential für eine größtmögliche Automatisierung der Arbeitsschritte zu bieten. Bzgl. der Digitalisierung werden für jedes Dokument, welches im Zuge eines Baugenehmigungsverfahrens geprüft wird, semantische (3D-)Modelle notwendig, die von Computerprogrammen interpretiert und ausgewertet werden können.
Die Automatisierung setzt die Digitalisierung und Abbildung auf semantische Modelle sämtlicher Arbeitsschritte (Prozessmodelle $\rightarrow$ BPMN), Entscheidungen (Entscheidungsmodelle $\rightarrow$ DMN), sowie der Eingangsdaten (Datenmodelle) voraus. Anhand der Prozessmodelle werden die digitalisierten Eingangsdaten sequentiell verarbeiten und über die Entscheidungsmodelle regel-basiert auf Gültigkeit und Zulassung überprüft.
Es lassen sich verschiedene Ebenen der Validierung definieren:

- **Formelle Ebene:** Validierung der Konformität, Vollständigkeit und Richtigkeit der Antragsdokumente
- **Konstruktionsebene:** Analyse des Bauwerks auf bautechnische Aspekte der Statik, des Brandschutz, Fluchtwege, etc.
- **Gebäudeebene:** Überprüfung des Bauwerks im lokaler Bereich bzgl. Grundstücksgrenzen, Abstandsflächen, Sichtschutz, Dienstbarkeiten, etc.
- **Stadtebene:** Überprüfung des Bauwerks auf regionaler Ebene bzgl. Regulierungen des Bebauungsplans, o.ä. (oder Ähnliches)

3 Das Datenmodell des digitalen 3D-Lageplans

3.1 Vorarbeit

3.1.1 Eingangsdaten für die Erstellung eines (amtlichen) Lageplans

Bevor ein Datenmodell für einen digitalen 3D-Lageplans entworfen werden kann, muss die abzubildende Objektmenge bestimmt werden. Um diese Objektmenge zu bestimmen, kann sich auf das geltende Gesetz (in diesem Fall der BauPrüfVO) NRW mit Stand vom 24.7.2020) der Bauvorlagen bezogen werden. Dieses besagt, dass *„der Lageplan auf der Grundlage eines Auszugs aus dem Liegenschaftskataster (§2) zu erstellen"* (BauPrüfVO NRW §3 Absatz 1 mit Stand vom 24.7.2020) ist. Des weiteren wird eine Liste an Objektarten aufgeführt, welche, soweit erforderlich, in einem Lageplan enthalten sein müssen.

Der Auszug aus dem Liegenschaftskataster in Deutschland liegt dem ALKIS zu Grunde. Im Zuge der Einführung von ALKIS entwickelte die AdV (Arbeitsgemeinschaft der Vemessungsverwaltungen der Länder der Bundesrepublik Deutschland) ein bundeseinheitliches Datenmodell, nach welchem die Daten des Liegenschaftskataster modelliert und bereitgestellt werden ("Eine Informationsbroschüre der Bayerischen Vermessungsverwaltung zur Einführung von ALKIS in Bayern", 2013). In dem Objektartenkatalog der AdV zum DLKM (Digitales Liegenschaftskataster-Modell) werden die Fachobjekte des DLKM auf Basis des AAA-Fachschemas aufgeführt.
Der Objektartenkatalog umfasst verschiedene Objektartengruppen:

- Angaben zum Flurstück
- Grundbuch
- Angaben zur Lage
- Angaben zum Netzpunkt
- Angaben zum Punktort
- Fortführungsnachweis
- Angaben zur Reservierung
- Angaben zur Historie
- Person- und Bestandsdaten
- Angaben zum Gebäude
- Tatsächliche Nutzung
- Siedlung

- Verkehr
- Vegetation
- Gewässer
- Bauwerke, Einrichtungen und sonstige Angaben
- Bauwerke, Einrichtungen in Siedlungsflächen
- Besondere Anlagen auf Siedlungsflächen
- Bauwerke, Anlagen und Einrichtungen für den Verkehr
- Besondere Vegetationsmerkmale
- Besondere Eigenschaften von Gewässern
- Besondere Angaben zum Gewässer
- Reliefformen
- Messdaten 3D
- Öffentlich-rechtliche und sonstige Festsetzungen
- Bodenschätzung, Bewertung
- Kataloge
- Geographische Gebietseinheiten
- Administrative Gebietseinheiten
- Angaben zu Nutzerprofilen
- Migrationsobjekte

(Pauly, 2019a)

Der Gesetzestext nach BauPrüfVO) NRW §3 Absatz 1 listet eine Beschreibung von Objektarten auf, die in einem Lageplan, soweit notwendig, darzustellen sind:

> *„(..)*
> *1. seinen Maßstab und die Lage des Baugrundstücks zur Nordrichtung,*
> *2. die Bezeichnung des Baugrundstücks und der benachbarten Grundstücke nach Straße, Hausnummer, Grundbuch und Liegenschaftskataster sowie die Angabe der Eigentümerin oder des Eigentümers des Baugrundstücks,*
> *3. die rechtmäßigen Grenzen des Baugrundstücks und deren Längen sowie seinen Flächeninhalt,*
> *4. die Höhenlage der Eckpunkte des Baugrundstücks und die Höhenlage des engeren Baufeldes bezogen auf das aktuelle amtliche Höhenbezugssystem,*

5. die Breite und die Höhenlage angrenzender öffentlicher Verkehrsflächen bezogen auf das aktuelle amtliche Höhenbezugssystem,
6. die vorhandenen baulichen Anlagen auf dem Baugrundstück und auf den angrenzenden Grundstücken sowie die genehmigten oder nach §63 Absatz 2 und 5 BauO NRW 2018 zulässigen, aber noch nicht ausgeführten baulichen Anlagen auf dem Baugrundstück, bei Gebäuden auch mit Angabe ihrer Geschosszahl, Wand- und Firsthöhen und deren Abstandflächen mit Berechnung,
7. Denkmäler im Sinne des Denkmalschutzgesetzes vom 11. März 1980 (GV. NRW. S. 226, ber. S. 716) in der jeweils geltenden Fassung auf dem Baugrundstück und dessen engerer Umgebung sowie geschützte Baumbestände auf dem Baugrundstück,
8. Flächen auf dem Baugrundstück, die von Baulasten betroffen sind, sowie Flächen auf den angrenzenden Grundstücken, die von Baulasten zugunsten des Baugrundstücks betroffen sind,
9. Flächen auf dem Baugrundstück, die mit grundbuchlich gesicherten Dienstbarkeiten zu Gunsten der Träger von Hochspannungsleitungen und unterirdischen Leitungen für die Versorgung mit Elektrizität, Gas, Wärme und Wasser belegt sind,
10. Hydranten und andere Wasserentnahmestellen für Feuerlöschzwecke,
11. die Bezeichnung des Bebauungsplanes oder anderer Satzungen nach dem Baugesetzbuch in der Fassung der Bekanntmachung vom 3. November 2017 (BGBl. I S. 3634) in der jeweils geltenden Fassung mit den Festsetzungen über Art und Maß der baulichen Nutzung, die Bauweise, die Darstellung der Baulinien und Baugrenzen und der Flächen auf dem Baugrundstück, für die der Bebauungsplan oder eine andere Satzung besondere Festsetzungen trifft, sowie die Bezeichnung der örtlichen Bauvorschriften,
12. die geplanten baulichen Anlagen unter Angabe der Außenmaße, der Dachform, der Wand- und Firsthöhen, der Höhenlage der Eckpunkte der baulichen Anlage bezogen auf das aktuelle amtliche Höhenbezugssystem an der Geländeoberfläche, der Höhenlage des Erdgeschossfußbodens bezogen auf das aktuelle amtliche Höhenbezugssystem, der Grenzabstände, der Tiefe und Breite der Abstandflächen, der Abstände zu anderen baulichen Anlagen,
13. die Abstände der geplanten baulichen Anlage zu öffentlichen Verkehrsflächen, zu Grünflächen, zu Wasserflächen und zu Wäldern,
14. die Aufteilung der nicht überbauten Flächen auf dem Baugrundstück unter Angabe der Lage, Anzahl und Größe der Stellplätze für Kraftfahrzeuge, der Fahrradabstellplätze, der Zu- und Abfahrten, der Bewegungsflächen für die Feuerwehr, der Kinderspielplätze und der Flächen, die gärtnerisch angelegt werden beziehungsweise mit Bäumen bepflanzt werden sollen sowie
15. die Lage der Entwässerungsgrundleitungen bis zum öffentlichen Kanal oder die Lage der Abwasserbehandlungsanlage mit der Abwassereinleitung."
(BauPrüfVO) NRW §3 Absatz 1 mit Stand vom 24.7.2020)

Zusätzlich wird vom BDVI ein Objektartenkatalog bereitgestellt, welche eine Zusammenstellung der wichtigsten und üblichsten darzustellenden Objekte in Lageplänen beinhaltet:

- Abstand Bauliche Anlage zu Bahnanlagen
- Abstand bauliche Anlage zu Grenzen
- Abstand Bauliche Anlage zu Heiden
- Abstand Bauliche Anlage zu Leitungen
- Abstand Bauliche Anlage zu Mooren
- Abstand Bauliche Anlage zu oberirdischen Gewässern
- Abstand Bauliche Anlage zu öffentlichen Grünflächen
- Abstand bauliche Anlage zu ortsfesten Behältern
- Abstand Bauliche Anlage zu Verkehrsflächen
- Abstand Bauliche Anlage zu Wäldern
- Abstand Bauliche Anlagen auf Nachbargrundstücken zu ihren Grundstücksgrenzen
- Abstand geplante bauliche Anlage zu Friedhöfen
- Abstand Hochspannungsleitungen zu baulichen Anlagen
- Abstand Leitungen zu geplanter baulicher Anlage
- Abstand zu Dämmen und Deichen
- Abstand zu oberirdischen Gewässern
- Abstände zu baulichen Anlagen
- Abstandsflächen
- Angabe Zuverlässigkeit Grenzen
- Angaben Erbbauberechtigter
- Angaben über andere Bestandteile von Natur und Landschaft nach §§23 bis 30 des Bundesnaturschutzgesetzes , „Natura 2000”-Gebiete nach §32 des Bundesnaturschutzgesetzes sowie Lebensstätten besonders geschützter Arten gemäß §7 Absatz 2 Nummer 13 des Bundesnaturschutzgesetzes und streng geschützter Arten gemäß §7 Absatz 2 Nummer 14 des Bundesnaturschutzgesetzes sowie Wald im Sinne des Bremischen Waldgesetzes
- Angaben zu nicht festgestellten Grenzen
- Anlagen zur Aufbewahrung von Exkrementen und Urin, jeweils auch mit Einstreu, sowie Gärresten

- Anzahl barrierefrei nutzbare Flächen außerhalb des Gebäudes
- Anzahl Stellplätze für Menschen mit Gehbehinderung, Rollstuhlnutzer
- Anzahl Stellplätze
- Anzahl Vollgeschosse
- Art der Außenwände
- Art der Bedachung
- Art und Maß der baulichen Nutzung
- Barrierefrei zugänglicher Haupteingang
- Bau- und Kulturdenkmale
- Baugrenze
- Bauherr
- bauliche Anlage Außenmaße
- Baulinie
- Bäume an öffentlichen Verkehrsflächen
- Bauweise
- Bepflanzte Flächen
- Bestätigung Vollständigkeit Gebäudebestand
- Bewegungsfläche Feuerwehr
- Bezeichnung Grundbuch Gemeinde
- Bezeichnung Grundbuch Grundbuchblatt
- Bezeichnung Grundbuch Hausnummer
- Bezeichnung Grundbuch Straße
- Bezeichnung Liegenschaftskataster Flur
- Bezeichnung Liegenschaftskataster Flurstücksnummer
- Bezeichnung Liegenschaftskataster Gemarkung
- BMZ (Baumassenzahl)
- Breite Abstandsflächen
- Breite öffentliche Verkehrsflächen

- Brunnen, Sicker- und Abfallgruben
- Dachform
- Datum Bauantrag
- Die gemäß §8 Absatz 2 des Bremischen Ausführungsgesetzes zum Bundesnaturschutzgesetz durch die Naturschutzbehörde beurteilten Angaben von Eingriffsvorhaben nach §17 Absatz 4 des Bundesnaturschutzgesetze
- Eigentümerangaben (Grundbuch)
- Entwässerungsgrundleitung bis zum öffentlichen Kanal einschließlich des Anschlusskanals und deren Nennweiten, die Reinigungsöffnungen und Schächte, sowie Kleinkläranlagen, Gruben, Abscheider oder Sickeranlagen mit deren Abwassereinleitung
- Erkennbarkeit von Grenzen in der Örtlichkeit
- Festsetzung Bebauungslan Erhaltungsgebote
- Festsetzung Bebauungslan Pflanzgebote
- Festsetzung Bebauungslan überbaubare Grundstücksfläche
- Festsetzungen eines Bebauungsplans für das Baugrundstück über die überbaubaren und die nicht überbaubaren Grundstücksflächen
- Festsetzungen zu Ausgleichs und Ersatzmaßnahmen
- Firsthöhe
- Flächen Abstandsflächenübernahmeerklärung
- Flächen des Baugrundstücks, die in einem Sanierungsgebiet oder im Geltungsbereich einer Erhaltungssatzung nach §172 BauGB (Baugesetzbuch) liegen
- Flächen des Baugrundstücks, die innerhalb des Gebiets einer Erhaltungssatzung liegen
- Flächen des Baugrundstücks, die innerhalb eines Sanierungsgebiets liegen
- Flächen des Baugrundstücks, die innerhalb Entwicklungsbereichs liegen
- Flächen des Baugrundstücks, die innerhalb des Geltungsbereichs einer Veränderungssperre liegen
- Flächen für die der Bebauungsplan besondere Festsetzungen trifft
- Flächen für Garagen und Stellplätze
- Flächen mit Baulasten
- Flächen mit grundbuchlich gesicherten Dienstbarkeiten
- Gebäudeklasse

- Gehwegüberfahrten
- Geplante bauliche Anlage
- Geschosszahl
- Geschützte Baumbestände
- Geschützte Landschafts-/ Naturbestandteile
- GFZ (Geschossflächenzahl)
- Grenzen
- Grenzlängen
- Größe barrierefrei nutzbare Flächen außerhalb des Gebäudes
- Größe Stellplätze
- Größe Stellplätze für Menschen mit Gehbehinderung, Rollstuhlnutzer
- GRZ (Grundflächenzahl)
- Hochspannungsleitungen
- Höhenlage EGFOK (Ergeschoss-Fußbodenoberkante)
- Höhenlage engeres Baufeld
- Höhenlage geplante bauliche Anlage
- Höhenlage Grenzpunkte
- Höhenlage öffentliche Verkehrsflächen
- Hydranten und Wasserentnahmestellen
- Hydranten und Wasserentnahmestellen Richtungs- und Entfernungsangabe
- In Planfeststellungsbeschlüssen ausgewiesene, noch nicht in einen Bebauungsplan übernommenen Verkehrsflächen
- Katastermäßige Flächengröße
- Kinderspielplätze (Anzahl)
- Kinderspielplätze (Fläche)
- Kinderspielplätze (Lage)
- Kronendurchmesser
- Lage Abwasserbehandlungsanlage
- Lage barrierefrei nutzbare Flächen außerhalb des Gebäudes

- Lage Entwässerungsgrundleitung
- Lage in einem Flurbereinigungsgebiet
- Lage in einem geschützten Grünbestand
- Lage in einem Grabungsschutzgebiet
- Lage in einem Landschaftsschutzgebiet
- Lage in einem Naturschutzgebiet
- Lage in einem Überschwemmungsgebiet
- Lage in einem Umlegungsgebiet
- Lage in einem Wald
- Lage in einem Wasserschutzgebiet
- Lage Stellplätze
- Lage Stellplätze für Menschen mit Gehbehinderung, Rollstuhlnutzer
- Leitungen (Wasser)
- Leitungen (Abwasser)
- Leitungen (Wärme)
- Leitungen (Elektrizität)
- Leitungen (Gas)
- Leitungen (Telekommunikation)
- Maßstab
- Maßstabsleiste
- Mit gesundheitsgefährdenden Stoffen belastete Flächen
- Nordrichtung
- Nutzung
- Öffentlicher Entwässerungskanal mit Sohlenhöhe und Rückstauebene
- Ortsfeste Behälter (Öl, Gas, brennbare Flüssigkeiten, wassergefährdende Flüssigkeiten)
- Plätze für Abfallbehälter
- Schächte, Entnahmestellen, Absperrvorichtungen der Versorgungseinrichtungen (Elektrizität, Wasser, Gas, Öl, Wärme)
- Stammumfang

- Straßengruppe öffentliche Verkehrsflächen
- Tiefe Abstandsflächen
- Vorhandene bauliche Anlagen
- Wald auf Baugrundstück
- Wandhöhen
- Wasserschutzzone ggf. Grenzverlauf
- Zu- und Abfahrten (Breite)
- Zu- und Abfahrten (Lage)
- Zugehörigkeit zu einer unter Denkmalschutz gestellten Gesamtanlage
- Zulässige noch nicht ausgeführte bauliche Anlagen

Zusammenfassend können thematische Bereiche von Objektarten bestimmt werden, die für die Erstellung eines Lageplans notwendig sind. Dies sind zum einen topographische Objekte, welche aus dem Bestand stammen, oder im Zuge des Bauvorhabens geplant sind. Zum anderen sind es konzeptionelle Objekte, wie rechtliche und administrative Konstrukte, die ebenfalls jeweils geplant sind oder Bestand haben.

Die Bestandsobjekte der topographischen Objektmenge stammen aus dem Auszug des Liegenschaftskatasters, oder anderen Liegenschaftsdatensätzen (z.B. das Leitungskataster von Stadtwerken), sowie aus Bestandsvermessungen (z.B. Laserscanning/Photogrammetrie für die Erhebung eines DGMs) des*der Vermessungsingenieur*in. Die geplanten baulichen topographische Objekte kommen aus dem Planungsmodell des*der Entwurfsverfasser*in.
Bestehende konzeptionelle Objekte, wie öffentlich-rechtliche Regulierungen und Festsetzungen bzgl. der Stadtplanung werden dem Bebauungsplan entnommen. Administrative Objekte, wie z.B. Flurstücke und Grundstücksgrenzen, Eigentümer, Dienstbarkeiten, Baulasten, etc., aus dem Liegenschaftskataster oder dem Baulastenverzeichnis. Konzeptionelle Objekte, wie z.B. die durch das Bauvorhaben resultierende Abstandsflächen, Teilung des Grundstücks, Grenzumlegung, etc., werden durch den*die Verfasser*in des (amtlichen) Lageplans erhoben.

In Abbildung 8 ist dargestellt, aus welchen Quellen die Daten für die Erstellung eines (amtlichen) Lageplans stammen. Dabei lässt sich die vorher beschriebene Einteilung der Daten in topographische Objekte (Bestand/geplant) und konzeptionelle Objekte (Bestand/geplant) nicht strikt auf die Datenherkunft anwenden. Bspw. (beispielsweise) wird im Liegenschaftskataster der Bestand topographischer Objekte (z.B. Baumbestand, Straßen,etc.) als auch konzeptioneller Objekte (z.B. Flurstücksgrenzen, Eigentümer, etc.) verwaltet. Ein Auszug des Liegenschaftskatasters beinhaltet somit beide Objektarten.

Datensätze, dessen Datenherkunft und thematische Art von Vornherein nicht sicher abschätzbar ist (z.B. Leitungskataster über die Lage von Versorgungsleitungen, DGM, o.ä.), werden unter dem Begriff ”Zusätzliche Datensätze” geführt.

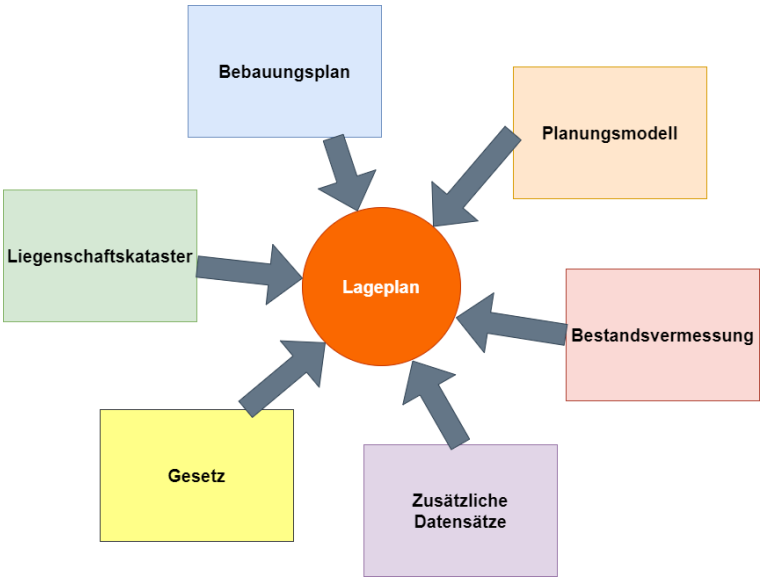


Abbildung 8 Eingangsdaten bei der Erstellung eines Lageplans.

3.1.2 Wahl eines Datenmodells als Basis für die Entwicklung des Datenmodells eines 3D-Lageplans

Die Erstellung eines 3D-Lageplans setzt ein Datenformat voraus, in dem die Dateninhalte nach einem unabhängigen Datenmodell eindeutig definiert werden. Dadurch kann gewährleistet werden, dass Datensätze zwischen verschiedenen Parteien der unterschiedlichen Phasen eines Bauvorhabens und deren Systemen vereinfacht ausgetauscht und verarbeitet werden können.

Es existiert eine Reihe verschiedener Datenmodelle aus den Bereichen der Geoinformation und AEC. Diese werden im Folgenden kurz vorgestellt und anhand ihrer Anwendbarkeit im konkreten Anwendungsfall des Lageplans in 3D bewertet. Anschließend wird darauf eingegangen, warum CityGML die größten Vorteile mit sich bringt und eine kurze Gegenüberstellung mit dem Datenmodell der Landesvermessungsämter durchgeführt.

Datenmodelle aus dem Bereich der Geoinformation und AEC

XPlanung

Das Datenmodell XPlanung unterstützt den Austausch von Bauleitplänen, Raumordnungsplänen und Landschaftsplänen (Brenner, 2019). Dadurch werden die verschiedenen Festsetzungen von Flächen des Bebauungsplans, welche auch im Lageplan bestand haben, eines Bereichs als objektstrukturierte Geodaten erfasst. Jedoch dient XPlanung prinzipiell der 2D-Darstellung, eignet sich also trotz (von CityGML übernommenem) ADE Konzept nicht für einen Lageplan in 3D (Brenner, 2019), da auch keine Gebäude im Modell berücksichtigt werden. Auch Flurstücke, welche zwingend in einem Lageplan erforderlich sind, sind in XPlanung nicht vorhanden. Trotzdem soll das XPlanung Modell in der Modellentwicklung für den Lageplan als Referenz dienen, um die im Lageplan enthaltenen Objekte des Bebauungsplans abzubilden.

LandInfra

Ein relativ neuer Standard aus dem Bereich des Bauingenieurwesens ist LandInfra.
LandInfra ist für die Abbildung von Infrastruktur in Bereichen von Umwelt- und Bauingenieurwesen konzipiert und dient bei der Planung von Projekten (Kumar, Labetski, Ohori, Ledoux & Stoter, 2019; Gilbert et al., 2020). Es wurde als Brücke zwischen der BIM- und GIS-Welt gestaltet, da es in Bereichen wie Geometrie und Semantik viele Überschneidungen mit den populären Standards CityGML (GIS) und IFC (BIM) aufweist (Kumar et al., 2019; Herle, Becker, Wollenberg & Blankenbach, 2020). Neben Merkmalen, die das Landschaftsbild bestimmen (Verkehrsnetze, Bauwerke, Gelände, etc.) wird auch die Landadministration und die Modellierung von Rechten bzgl. Landparzellen berücksichtigt (Kumar et al., 2019). Im Vergleich zu CityGML ist es bei dem LandInfra-Datenmodell nicht möglich bspw. ein Gebäude in seine semantischen Flächen, wie z.B. Wandfläche oder Dachfläche, zu zerlegen. Auch ist keine Unterstützung für verschiedene LOD vorhanden (Gilbert et al., 2020). Trotz der Überlappung einiger thematischer Bereiche ist der Arbeitsbereich von LandInfra mehr auf Schienen- und Straßeninfrastruktur ausgelegt und adaptiert Konzepte des LADM aus den Bereichen Vermessung und Kataster (Gilbert et al., 2020). Aufgrund des geringen Detaillevels bei der semantischen Darstellung von Gebäuden durch begrenzende Flächen ist LandInfra nicht für die 3D-Lageplan-Modellierung geeignet. Durch die kürzliche Aufnahme von LandInfra als OGC (Open Geospatial Consortium)-Standard im Jahr 2016 und dessen Datenformat InfraGML ein Jahr später, gibt es außerdem bis jetzt wenig Softwareunterstützung für LandInfra-Datensätze (Kumar et al., 2019).

Industry Foundation Classes

Aus dem Bereich des BIM stehen Datenmodelle, wie der offene Standard IFC, zur Verfügung. Diese Datenmodelle haben ihre Kernkompetenz jedoch auf der Abbildung, dem Planen und Design von Bauprozessen und Bauwerken auf Bauwerksebene, und besitzen dementsprechend einen sehr hohen Detailgrad.

BIM-Modelle sind so konzipiert, dass sie bautechnisch relevante Elemente abbilden können und detailgetreu modellieren, wie das Bauwerk aus Konstruktionsteilen zusammengesetzt ist (Ohori, Biljecki, Kumar, Ledoux & Stoter, 2018). Jedoch sind diese Elemente für die Darstellung eines Lageplans nicht notwendig. Nach §7 der BauPrüfVO sind in Lagepläne keine bautechnischen Elemente des Gebäude selbst, sondern vielmehr der Bauwerksumriss bzw. die Bauwerkshülle, mit einhergehende rechtliche Objekte, Umgebungsobjekte und Nachbarschaften des Bauwerks einzutragen. Die Grundlagendaten stammen also großteils von Vermessungsarbeiten, die die von außen erfassbare Topographie von Objekten dokumentieren. Eine Bestandsdokumentation ist zwar auch mit BIM möglich (z.B. in Effkemann (2019); Hellmann (2019)), für die Zusammenführung von Gebäudedaten mit Daten aus dessen Umgebung, wie z.B. einem DGM, ist das Datenmodell des BIM jedoch aktuell noch nicht praktikabel (Becker et al., 2019). Zusätzlich ist IFC nicht ohne weiteres für individuelle Nutzeranwendungen erweiterbar, d.h. das bestehende Modell kann nicht um fehlende Klassen ergänzt werden, wie das z.B. bei CityGML über das ADE-Konzept möglich ist (Schönhut, 2018). Es müsste also für jeder Objekttyp, der im Lageplan enthalten sein kann, auf eine entsprechende Klasse im IFC-Datenmodell abgebildet werden, die die Eigenschaften befriedigend abdecken kann. Ist dies nicht der Fall, können die nötigen Eigenschaften nur mit *IfcPropertySets, IfcProxy* eingebracht werden, oder das Schema muss erweitert werden (André Borrmann & Liebich, 2015; Rumor, Coors, Fendel & Zlatanova, 2007; *Die Eigenschaften der IFC-Objekte: IfcPropertySet*, o. J.; Rajabifard, Atazadeh & Kalantari, 2019; Weise, Liebich & Wix, 2008). Durch diese nicht standardisierte Form der Erweiterung des IFC-Schemas kann es jedoch zu unerwünschten Komplikationen bei der Kompatibilität von verschiedenen Datensätzen kommen (Weise et al., 2008).

INSPIRE

Das Datenmodell INSPIRE hat zum Ziel, Geodaten verschiedener Themenbereiche europaweit zu harmonisieren und bereitzustellen (*About INSPIRE*, 2021). Aufgrund dieser europaweiten Ausrichtung des Datenmodells wurde es so generell wie möglich gehalten, um es mit den verschiedenen nationalen Standards der unterschiedlichen Mitgliedsstaaten kompatibel zu halten. Das INSPIRE-Datenmodell beinhaltet insgesamt 34 räumlicher Themengebiete, was im Umfang über die Anforderungen beim Lageplan hinausgehen, im Detail jedoch zu generell gehalten ist. Für eine Anwendung des INSPIRE-Modells bei der Modellierung des Lageplans fehlt aufgrund der Zielsetzung der räumlichen Reichweite des INSPIRE-Standards die nötige Informationstiefe, nationales Recht und Administration im Detail darzustellen. Dies ist jedoch im Fall von amtlichen Lageplänen notwendig, da diese an lokale Gesetze und Festsetzungen gebunden sind.

AFIS-ALKIS-ATKIS

Im öffentlichen Sektor wurde mit dem AAA-Modell ein bundesweit einheitliches Datenmodell für die Aufnahme, Bereitstellung und Harmonisierung von amtlichen Daten des ALKIS, des ATKIS (Amtliches Topographisch-Kartographisches Informationssystem) und des AFIS (Amtliches Festpunktinformationssystem) etabliert (Schüttel, 2009).

Generell fällt die Erstellung eines Lageplans in den amtlich öffentlichen Bereich, sodass es sinnvoll erscheint, das dreidimensionale Lageplan-Konzept in das amtliche Datenmodell einzubetten und gegebenenfalls durch notwendige Klassen zu erweitern. Da dem Lageplan ein Katasterauszug zugrunde liegt (BauPrüfVO NRW §2 mit Stand vom 24.7.2020), könnte direkt auf den vom Land bezogenen amtlichen Daten aufgebaut werden.
Laut dem AAA-Fachschema ist es möglich, 3D-Gebäude bis zu LOD3 abzubilden (Pauly, 2019b; "Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok)", 2018). Die *AX_Bauteil3D*-Klasse stellt dazu das Attribut "gehörtZu" vom Typ *AX_Gebäude* bereit. *AX_Bauteil3D* wird durch Begrenzungsflächen (Wand-, Dach-, Boden,- Abschlussfläche) mit Öffnungen (Türen und Fenster) definiert und kann Gebäudeinstallationen besitzen. Dieser Modellierungsansatz von 3D-Gebäude ähnelt stark dem Konzept von CityGML. Um dem Produktstandard für 3D-Geäudemodelle der AdV zu entsprechen wurde das AdV-CityGML-Profil eingeführt, das das ursprüngliche CityGML-Modell der Version 1.0 dementsprechend einschränkt (Aringer, 2016). Insbesondere werden nur die Module *Building, Generics* und *Apperance* zugelassen (Aringer, 2016). Auch die Attribute und Assoziationen innerhalb der zugelassenen Module wurden gemäß des Produktstandards der AdV geändert (Aringer, 2016).
Das AAA-Basisschema, woraus sich das AAA-Fachschema ableitet, kann beliebig mit anderen Fachschemata erweitert werden, oder das AAA-Fachschema selbst als Erweiterungsgrundlage verwendet werden (Schüttel, 2009; Seifert, 2005). So könnten dem Datenmodell fehlende Klassen hinzugefügt werden.

CityGML

CityGML ist ein Plattform-unabhängiges XML-basiertes Format zum Austausch von virtuellen 3D-Stadtmodellen und implementiert ein Anwendungsschema der GML (Geographic Markup Language) (Gröger et al., 2012). Neben der 3D-Geometrie und Informationen über die Visualisierung der Objekte bietet CityGML zudem reichhaltige semantische und topologische Information (Gröger et al., 2012).
CityGML ist ein geographisches Informationsmodell für den urbanen Kontext (Gröger et al., 2012). Es bietet neben dem geometrischen Grundkonzept verschiedene Module, die die topographischen Objekte in thematische Bereiche einteilt. Außerdem ist es mit CityGML möglich, Objekte simultan mit verschiedenen geometrischen Repräsentationen zu instanziieren (Gröger et al., 2012). Dies wird durch das LOD-Konzept möglich. Ein weiteres Merkmal ist der Modellerweiterungsmechanismus ADE, welcher es erlaubt, das CityGML-Basismodell anwendungsspezifisch zu erweitern (Gröger et al., 2012).
Für den Anwendungsfall des 3D-Lageplans bietet das Modell des CityGML-Standards die Möglichkeit der 3D-Modellierung, sowie die Klassen der meisten topographischen Objekte im urbanen Umfeld (Gröger et al., 2012). Die Klasse der generischen Objekte (GenericCityObject) macht das Modell flexibel anpassbar (Gröger et al., 2012). Auch kann auf den ADE-Mechanismus zurückgegriffen werden, welches erlaubt, bzgl. des Anwendungsfalls "Lageplan" dem CityGML-Modell neue Klassen hinzuzufügen.
Weitere Eigenschaften des CityGML-Standards, die bei der Modellierung eines Lageplans von Vorteil sind, werden im folgenden Abschnitt näher betrachtet.

CityGML

Vorteile gegenüber anderen Standards

Gegenüber den bereits genannten Standards bietet das CityGML-Datenmodell einige Vorteile. Im Gegensatz zu BIM, das sich auf das Bauwerk, und das mit hohem Detailgrad, konzentriert, ist das CityGML-Modell großräumig skalierbar und kann somit auch die Nachbarschaft des Gebäudes und andere Objekte in der Peripherie abbilden (siehe Abbildung 9) (*Basic Information*, 2015). Durch verschiedene geometrische Detailstufen, bis zu architektonischem Grad (*Basic Information*, 2015; Brüggemann & von Both, 2015), kann ebenfalls ein hoher Grad an Details abgebildet, gleichzeitig für den Lageplan überflüssige Information weggelassen werden.

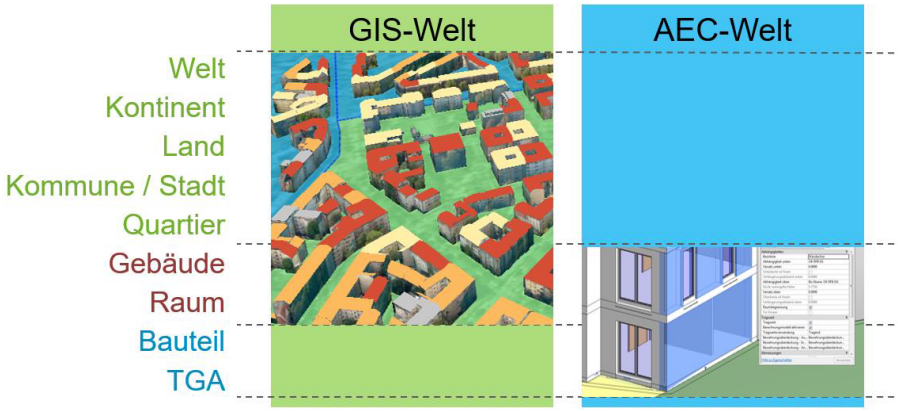


Abbildung 9 Unterschiede der Skalenbereiche zwischen GIS und BIM (Kaden et al., 2019).

Ein weiteres Argument für die Nutzung von CityGML ist, dass das CityGML-Modell wegen des Erweiterungskonzepts (ADE) relativ einfach auf den Anwendungsfall des Lageplans angepasst werden kann. Das ADE-Konzept ermöglicht die Erweiterung der vorhandenen Klassen des CityGML-Modells um zusätzliche domänen-spezifischen Attribute (Gröger et al., 2012). Auch kann das CityGML-Modell mit neuen Klassen, die sich von vorhandenen CityGML-Klassen ableiten, ergänzt werden (Gröger et al., 2012). Mit Hilfe der Klasse *GenericCityObject* kann zudem flexibel auf nicht im Datenmodell vorhandene Klassen reagiert werden und diese Objekte zur Laufzeit auf die generischen Objekte abbilden (Gröger et al., 2012).
Des weiteren bringt der CityGML-Standard bereits viele Klassen mit sich, die den urbanen Kontext abbilden, sodass auf viele der vorhandenen Klassen zurückgegriffen werden könnte. Zusätzlich ist dieser Standard schon vielfach in Verbindung mit dem Bau- und Architektursektor gebracht worden. Diesbezüglich existieren bspw. Forschungsarbeiten, die sich mit der Konvertierung von IFC nach CityGML, bzw. von CityGML nach IFC beschäftigen (Salheb, 2019; Donkers, Ledoux, Zhao & Stoter, 2015). Weitere Veröffentlichungen wie von Floros, Ellul und Dimopoulou (2018), sowie von El-Mekawy und Östman (2010) handeln ebenfalls von der semantischen Abbildung zwischen IFC und CityGML. CityGML stellt damit das Verbindungsstück zwischen der GIS- und BIM-Welt dar (Tobiáš, 2015).

Ein weiterer Vorteil von CityGML ist, dass die Möglichkeit besteht, die Objektgeometrien in verschiedenen LODs bereitzustellen. Dadurch kann in einer geeigneten Software von einem 3D-Modell des Lageplans zu einem 2D-Modell gewechselt werden. Es bietet sich an, Berechnungen wie Abstände oder Flächeninhalte (z.B. die zulässige Grundfläche des geplanten Gebäudes) im 2D-Modell durchzuführen, Volumenberechnungen (z.B. die zulässige Baumasse des geplanten Gebäudes) hingegen im 3D-Modell.

Gegenüberstellung der Datenmodelle AFIS-ALKIS-ATKIS und CityGML

In Abbildung 10 ist abgebildet, wie die thematische Abdeckung der Datenmodelle AAA und CityGML bzgl. des Objektkataloges des BDVIs eingestuft wird. Dabei wurde auch die mögliche Dimension der darstellenden Geometrie der Objekte berücksichtigt. Da der BDVI die Lagepläne in Deutschland nach dem Recht des jeweiligen Bundeslandes anfertigt, ist es nicht verwunderlich, dass die Abdeckung zwischen AAA und dem Objektkatalog des BDVIs sehr hoch ist. V.a. im Bereich der öffentlichen Vermessungs- und Planungstätigkeit ist folglich eine gute Abdeckung zu erkennen. Die übrigen Themengebiete lassen sich von beiden Datenmodellen gleichermaßen mehr oder weniger abbilden. Bemerkenswert ist, dass trotzdem, dass CityGML ein internationaler Standard ist (Löwner et al., 2013), es erstaunlich gut die Anforderungen und Objekte des BDVIs bedient. Dies lässt sich damit begründen, dass sich die Entwicklungen der beiden Datenmodelle gegenseitig beeinflusst haben (Löwner et al., 2013). Es ist ersichtlich, dass das AAA-Fachschema prinzipiell nicht für die 3D-Modellierung ausgelegt ist, obwohl im AAA-Basisschema mit dem Paket *AAA_Unabhaengige Geometrie 3D* Klassen für die 3D-Geometrie bereitstehen. Die Stärke von CityGML liegt eindeutig in der Konzeptionierung auf die 3D-Modellierung, weist dafür Lücken im öffentlich-/privat-rechtlichen und administrativen Bereich auf. Es ist zudem zu erkennen, dass beide Standards die baurechtlichen Objekte in ihrer Modellierung nicht berücksichtigen.
Da im Anwendungsfall des Lageplans in 3D die 3D-Modellierung einen fundamentalen Bestandteil einnimmt, wird hier CityGML dem AAA-Modell vorgezogen, auch wegen der erleichterten Erweiterung des Datenmodells durch das ADE-Konzept und der generischen Objekte. Zudem ist CityGML, national wie international, sehr weit verbreitet. Dementsprechend bieten viele Softwareprodukte Schnittstellen zum CityGML-Datenmodell an (*Software systems that provide CityGML support*, 2019).

Ein weiterer Aspekt, der für die Verwendung von CityGML spricht, ist die neue Version CityGML 3.0. Grundlage für das neue Konzept von CityGML 3.0 ist die Aufteilung der Welt in volumetrische Objekte und flächenhafte Objekte, welche als Begrenzungen für die volumetrischen Objekte dienen (Kutzner, Chaturvedi & Kolbe, 2020). Im Zuge der Entwicklung von CityGML 3.0 wurde auch die Kompatibilität zu anderen Standards wie IFC berücksichtigt (Kutzner et al., 2020). Die neue Version beinhaltet, neben Änderungen bestehender Module und weiterer neuer Module, das Modul *Construction*. Diese Klasse ist eine Unterklasse der Klasse *AbstractOccupiedSpace*, welche ein Volumenobjekt modelliert. Von der *AbstractConstruction* Klasse werden alle Bauwerksklassen wie *Building, Tunnel, etc.* als Subklassen abgeleitet.

Abbildung 10 Thematische Abdeckung der Datenmodelle AAA-Fachschema und CityGML bezogen auf den Objektkatalog des BDVIs.

Wenn bspw. ein *IfcWall*-Objekt des BIM-Standards IFC nach CityGML abgebildet werden soll, kann die Klasse *BuildingConstructiveElement* verwendet werden. Diese ist als Unterklasse der Klasse *AbstractConstructiveElement* eine Subklasse der Volumenobjekt-Klasse *AbstractOccupiedSpace*. *BuildingConstructiveElement* als Subklasse von *AbstractConstructiveElement* erbt das Attribut vom Typ *AbstractThematicSurface*. Im Fall eines *IfcWall*-Objekts könnte das Volumen mittels der CityGML-Klasse *WallSurface* des Construction-Moduls durch Flächenelemente begrenzt werden.

Ein zusätzlicher Argumentationspunkt für die Verwendung des CityGML Standards ist das GeoBIM Projekt der Technischen Universität Delft in Zusammenarbeit mit EuroSDR (European Spatial Data Research) (siehe Abschnitt 2.2.1). Ziel des Projekts ist die internationale Vereinheitlichung der Integration von Geoinformation und BIM. Dabei spielt in den jeweiligen Veröffentlichungen (de Laat & van Berlo, 2010; Noardo et al., 2019a, 2019b) CityGML die Rolle der Schnittstelle zwischen BIM und Geoinformation. Um die internationale Standardisierung voranzutreiben ist es daher sinnvoll, sich bei der Modellierung auf das im GeoBIM Projekt vorgeschlagenen Datenmodell für Stadtmodelle (CityGML) zu beziehen.

Auch im Projekt ”BIM-basierter Bauantrag” (siehe Abschnitt 2.2.2) wird der Lageplan im GML-Datenformat gefordert (siehe Abbildung 7). Welches Anwendungsschema genau gemeint ist, bleibt offen. Jedoch wäre der CityGML-Standard als Verbindungsstück zwischen GIS und BIM (siehe Abschnitt 3.1.2) die logische Konsequenz.

Entwicklung einer CityGML Application Domain Extension

Im Fall der Version CityGML 2.0 ist sich bei der Entwicklung einer ADE an die vom OGC als *Best Practice* angesehene Vorgehensweise der UML-Modellerweiterung und XSD (XML Schema Definition)-Schema-Ableitung mittels des Werkzeugs *ShapeChange* zu halten (van den Brink, Stoter & Zlatanova, 2014). Dabei wird zunächst versucht, die Domänen-spezifischen Klassen auf vorhandene CityGML-Klassen abzubilden, um von diesen dann gleichnamige Subklassen abzuleiten, die mit dem Stereotype *«ADEElement»* versehen werden (van den Brink et al., 2014). Die Spezialisierung wird optional mit dem Stereotyp *«ADE»* gekennzeichnet. Ist eine neue noch nicht vorhandene Klasse erwünscht, wird diese als entsprechende neue Subklasse der jeweiligen Superklasse modelliert und mit dem Stereotype *«FeatureType»* gekennzeichnet (van den Brink et al., 2014).
Die ADE wird auf Grundlage des aktuellen *Enterprise Architect*-Projekts modelliert und mittels *ShapeChange* ein XML-Schema abgeleitet.

Ab CityGML 3.0 stehen Klassen wie z.B. *ADEOfAbstractLogicalSpace, ADEOfBuilding*, etc. zur Verfügung, durch die über Ableitung von Subklassen ebenfalls ein Zugangspunkt (”ADE-Hook”) für die ADE-Entwicklung geschaffen wurde (`https://github.com/opengeospatial/CityGML-3.0CM/blob/master/ConceptualModel/CityGML_3.0_UML-Diagrams.pdf`). Diese sollen verwendet werden, um der jeweiligen Klasse Attribute hinzufügen zu können, ohne extra eine zusätzliche Unterklasse erstellen zu müssen. Ist dieser neue Mechanismus nicht ausreichend, muss auch in CityGML 3.0 eine Subklasse von einer existierenden CityGML-Superklasse abgeleitet werden.

Wahl von CityGML 3.0 als Basismodell für die Entwicklung des Lageplan-Datenmodells

Wie bereits geschildert, führt die neue Version 3.0 des CityGML-Standards zu einer Verbesserung der interdisziplinären Kompatibilität, sowie der Interoperabilität zwischen Standards unterschiedlicher Domänen (v.a. zu AEC).
Auch wenn zum Zeitpunkt der Erstellung der Arbeit CityGML 3.0 noch nicht als offizieller Standard anerkannt ist, werden die Vorteile der Neuerungen im Vergleich zu der Vorgängerversion CityGML 2.0 als so gewinnbringend angesehen, dass CityGML 3.0 v.a. im Bereich der GIS/BIM-Integration, nach der Veröffentlichung, schnell Einzug erhalten müsste.
Durch die in Abschnitt 2.2 geschilderten Rollen des Lageplans in den Baugenehmigungsverfahren, sowie dem dabei vorgesehenen Anwendungsfeldern und dem Austausch zwischen den Fachbereichen von GIS und BIM, wird deutlich, dass die Entwicklung des Lageplan-Modells auf Basis des CityGML 3.0-Modells stattfinden sollte, um eine effiziente und verlustlose Interoperabilität zu gewährleisten.

3.1.3 Anforderungen an das 3D-Lageplan-Datenmodell

Im Voraus werden bestimmte Anforderungen an das Datenmodell eines 3D-Lageplan gestellt. Konkret soll das Datenmodell

1. eine 3D-Repräsentation der Lageplan-Objekte unterstützen,
2. Kompatibilität und Interoperabilität zu anderen (internationalen) existierenden Standards aufweisen,
3. die Möglichkeit bieten, dem Instanzen-Dokument einen offiziellen und rechtlichen Charakter zu verleihen,
4. Vollständigkeit der im Kontext des Lageplans abzubildenden Objektklassen aufweisen,
5. es möglich machen, einen 2D-Lageplan von dem 3D-Lageplan abzuleiten,
6. die Möglichkeit bieten, Analysen und Berechnungen durchzuführen,
7. verschiedene Anwendungsfelder innerhalb und außerhalb des Bereichs des Bauvorhabens berücksichtigen,
8. unterschiedliche Varianten des Lageplans instanziierbar machen.

Zu 1.: 3D-Repräsentation der Lageplan-Objekte

Um eine besseren Eindruck des Bauvorhabens im Kontext der Umgebung zu schaffen, ist es notwendig die Szenerie in 3D zu erzeugen. Wechselwirkungen und Kollisionen von topologischen Objekten und konzeptionellen Sachverhalten können dadurch auch in 3D abgeschätzt werden. Auch kann die Bürgerbeteiligung durch die Bereitstellung von 3D-Visualisierungen des Projekts erhöht werden, was höhere Akzeptanz und Informiertheit unter der Bevölkerung zur Folge hat (Egger, 2019).
Auch wird bei der Prüfung des Lageplans im Baugenehmigungsverfahren durch dynamische 3D-Ansichten und v.a. semantische Information der Lageplan-Objekte das Verständnis von Berechnungen (z.B. von Abstandsflächen, GRZ, GFZ, BMZ) gesteigert.

Zu 2.: Kompatibilität und Interoperabilität zu anderen (internationalen) existierenden Standards

Wie in Abbildung Abbildung 8 dargestellt, sind die für die Erstellung eines amtlichen Lageplans zwei Datensätze aus dem öffentlichen Bereich notwendig (zusätzlich zu anderen Daten und Informationen wie Leitungspläne, DGM, etc.). Dies ist zum einen die Flurkarte, zum anderen der Bebauungsplan. Für beide Thematiken bestehen nationale und internationale Standards, die auf GML basieren. Für Deutschland steht für die Daten des Liegenschaftskatasters das AAA-Fachschema zur Verfügung, auf europäischer Ebene existiert das INSPIRE-Thema *CadastralParcel*. Für öffentlich Pläne wie den Bebauungsplan wurde XPlanung auf deutscher Ebene etabliert. Die europäische Korrespondenz hierfür stellt das INSPIRE-Thema *LandUse* dar. Da die europäischen Datenmodelle von INSPIRE und das LADM Ansätze für die Modellierung der genannten thematischen Bereiche auf internationaler Ebene bereitstellen, sollen sie als Grundlage der Modellierung der CityGML-Lageplan-ADE dienen, um Interoperabilität zwischen internationalen Standards gewährleisten zu können.

LADM wird als Referenz dienen, um rechtliche Beziehungen, zwischen verschiedenen Parteien in Bezug auf räumlichen Objekten zu modellieren (Lemmen, van Oosterom & Bennett, 2015). In J. Paasch, Oosterom, Paulsson und Lemmen (2013); J. M. Paasch, van Oosterom, Lemmen und Paulsson (2015) wird eine spezifische Erweiterung des LADM Legal Profiles um öffentliche bzw. private Rechte, Verpflichtungen und Einschränkungen vorgestellt, welche den Bezug zwischen dem LADM und dem LCDM (Legal Cadastral Domain Model), beschrieben in (J. Paasch, 2012), herstellt (J. Paasch et al., 2013). Dies kann verwendet werden um Grunddienstbarkeiten, sowie Baulasten, etc. zu modellieren. Das LADM bringt das Konzept von RRR (Right-Restriction-Responsibility) mit einem Eigentumsobjekt geographischer Ausdehnung in Verbindung.
LADM erlaubt die administrative Abbildung der Rechte, Verbote und Verpflichtungen an Eigentum (in diesem Fall Grundstück) mit räumlicher Ausdehnung gegenüber privaten, wie öffentlichen Parteien. Allerdings berücksichtigt das Modell nur einzigartige und homogene Rechte (*"unique and homogenous rights"*), d.h. das gesamte Rechtsobjekt (Grundstück) ist von dem Recht betroffen (*D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines*, 2014; "Guidelines on Real Property Units and Identifiers", 2004).
Rechtliche Belastungen wie Grunddienstbarkeiten (z.B. die Nutzung eines Weges) oder öffentlich-rechtliche Baulasten beziehen sich auf die Ganzheit des Rechtsobjekts (Grundstück), obwohl möglicherweise nur Teilflächen aktiv betroffen sind. In Lemmen, Oosterom und Netherlands (2010) werden verschiedene Lösungsansätze zur Modellierung von rechtlichen Objekten, die nur Teile des Rechtsobjekts betreffen, innerhalb des LADM erläutert.
Konzepte für öffentliche Festsetzungen der Landnutzung in der Planung, wie im Falle eines Bebauungsplans, werden im LADM *Spatial Planning Information Package* vorgestellt (Lemmen et al., 2019; Indrajit, van Loenen, Ploeger & van Oosterom, 2020). Das LADM-Paket wurde u.a. unter Berücksichtigung des INSPIRE-Themas *LandUse* entwickelt, welches erlaubt geplante Landnutzung abzubilden (Lemmen et al., 2019; Indrajit et al., 2020). Damit sind die Klassen beider Schemata konsistent und lassen sich ineinander überführen (z.B. LADM::SP_PlanningUnit $\leftrightarrow$ INSPIRE::ZoningElement, als Repräsentation von sektoriellen Festsetzungen; LADM::SP_PlanningBlock $\leftrightarrow$ INSPIRE::SpatialPlan, etc.).
Im deutschsprachigen Raum werden in Form des Katasters der ÖREB (Öffentlich-rechtliche Eigentumsbeschränkung) Überlegungen und Forschungsarbeiten angestellt, die öffentlich-rechtlichen Festsetzungen mit den Kataster-Daten zusammenführen (Aringer, 2017).

Die Modellierung der planerischen Festsetzungen des Bebauungsplans wird über die in Abbildung 11 dargestellten Ebenen von internationaler zu nationaler Ebene durchgeführt. Die Verknüpfung des CityGML-Modells mit der konzeptionellen Domäne der räumlichen Planung wird durch das Paket *Spatial Planning Information Package* des LADMs realisiert. An das *Spatial Planning Information Package* wird auf europäischer Ebene das INSPIRE-Modell *Planned Land Use* angebunden. Um die nationalen Details der Bebauungspläne abbilden zu können, werden auf nationaler Ebene die Klassen des XPlanung-Modells für Bebauungspläne von den INSPIRE-Klassen abgeleitet.

Für Erweiterung des CityGML-Standards um den Bereich der Landadministration und des Katasters dienen die Kernmodelle des LADM-Standards. An diese Modelle wird das Modell des INSPIRE-Themas *CadastralParcel* angeschlossen und mit Klassen des AAA-Fachschemas spezifiziert. Von Vorteil ist, dass das Kataster-Schema von INSPIRE auch hier mit dem des LADM konsistent ist und sich die Schemata somit leicht verknüpfen, bzw. voneinander ableiten lassen (Seifert, 2012).
Die Klassen der nationalen Datenmodelle werden an die internationalen Klassen des INSPIRE-Modells bzw. des LADM-Modells geknüpft, damit sowohl Kompatibilität zu nationalen, sowie internationalen Standards gewährleistet wird.
Für die Interoperabilität von Daten zwischen verschiedenen Domänen ist es notwendig, die Datenmodellierung auf existierende Standards zu stützen.

Um die Konformität der Lageplan ADE zu internationalen Standards zu gewährleisten wird die ADE im administrativen und Land-planerischen Bereich in verschiedene Ebenen (*Layer*) eingeteilt, die den administrativen Ebenen und daher der Reichweite des Standards entsprechen (siehe Abbildung 11). Die oberste Ebene ist der CityGML-Standard an dessen Klassen sich die ADE ”anhängt”. An die CityGML-Klassen werden Klassen der internationalen Standards LADM und INSPIRE (Gültigkeitsbereich ist die EU (Europäische Union)) geknüpft. Die nationalen Standards wie XPlanung und ALKIS werden anschließend an die Klassen des LADM-Standards bzw. INSPIRE-Standards angehängt. Im Fall von XPlanung wird sich dabei auf die Abbildungsregeln in Duan und Benner (2019) bezogen.

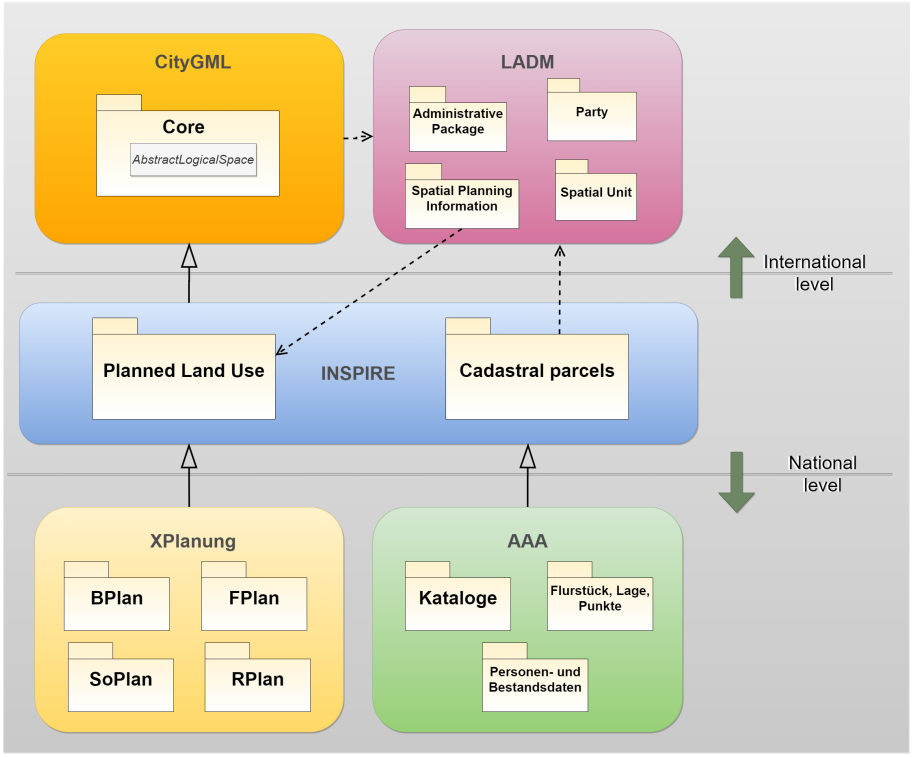


Abbildung 11 Modellierungslayer von internationaler Ebene zu nationaler Ebene. Die verschiedenen Standards, welche für das Lageplan-Modell fusioniert werden, weisen bereits wechselseitige Einflüsse und Beziehungen untereinander auf.

Zu 3.: Offizieller Charakter und rechtliche Anforderungen

Eine zusätzliche Anforderung ist, bestimmte Attribute, die für die rechtliche Haftbarkeit notwendig, sind im Datenmodell des Lageplans zu berücksichtigen. Dies ist notwendig, da ein amtlicher Lageplan ein beglaubigtes Urkundendokument darstellt. Um den Ursprung der im Lageplan dargestellten Daten nachvollziehbar zu machen, und Rechtssicherheit zu schaffen, muss dies in Attributen der Lageplan-Objekte festgehalten werden. Bspw. müssen "nachrichtlich übernommene" Daten, die von Drittanbietern bereitgestellt werden (z.B. Pläne von Leitungsverläufen der Stadtwerke) und nicht vermessungstechnisch überprüft werden können, gekennzeichnet werden. Genauso sind die von dem*der Ersteller*in des amtlichen Lageplans erhobenen Daten (z.B. Abstandsflächen, neu vermessene Grenzen, etc.) entsprechend zu kennzeichnen.
Auch werden für die Übersichtlichkeit Referenzen zu den Beteiligten (Bauherr, Planungsbüro, Vermessungsbüro, Baubehörde, etc.) im Bauvorhaben, sowie zum globalen Bauantrag notwendig. Auch müssen die Rechenschritte von Berechnungen von GRZ, GFZ, BMZ oder Abstandsflächen in Attributen des jeweiligen Objektes festgehalten werden, um Klarheit und Nachvollziehbarkeit bei der Überprüfung der Werte zu gewährleisten. Um so viel Transparenz und Nachvollziehbarkeit der Daten und ihrer Herkunft herzustellen, sind deshalb Metadaten notwendig:

- Datenursprung (z.B. übernommene Daten oder selbst erhobene Daten, Behörde, Vermessungs-/Planungsbüro)
- Qualitätsangabe (z.B. Genauigkeit)
- Verarbeitungsschritte (z.B. Filterung der Daten, Bereitstellung der Berechnungsschritte)
- Art der Datenerhebung (z.B. Messmethode)
- Beteiligte im Bauvorhaben
- Referenz zu Bauantrag

Um das XML-Dokument eines Lageplans elektronisch rechtskräftig wirksam machen zu können, muss es möglich sein, der Datei eine qualifizierte elektronische Signatur beizufügen. Im SigG (Signaturgesetz) §2 werden elektronische Signaturen wie folgt kategorisiert:

> *„Im Sinne dieses Gesetzes sind*
>
> 1. *"elektronische Signaturen" Daten in elektronischer Form, die anderen elektronischen Daten beigefügt oder logisch mit ihnen verknüpft sind und die zur Authentifizierung dienen,*
> 2. *"fortgeschrittene elektronische Signaturen" elektronische Signaturen nach Nummer 1, die*
>
>   a) *ausschließlich dem Signaturschlüssel-Inhaber zugeordnet sind,*
>
>   b) *die Identifizierung des Signaturschlüssel-Inhabers ermöglichen,*

c) *mit Mitteln erzeugt werden, die der Signaturschlüssel-Inhaber unter seiner alleinigen Kontrolle halten kann, und*

d) *mit den Daten, auf die sie sich beziehen, so verknüpft sind, dass eine nachträgliche Veränderung der Daten erkannt werden kann,*

3. *"qualifizierte elektronische Signaturen" elektronische Signaturen nach Nummer 2, die*

   a) *auf einem zum Zeitpunkt ihrer Erzeugung gültigen qualifizierten Zertifikat beruhen und*

   b) *mit einer sicheren Signaturerstellungseinheit erzeugt werden."*

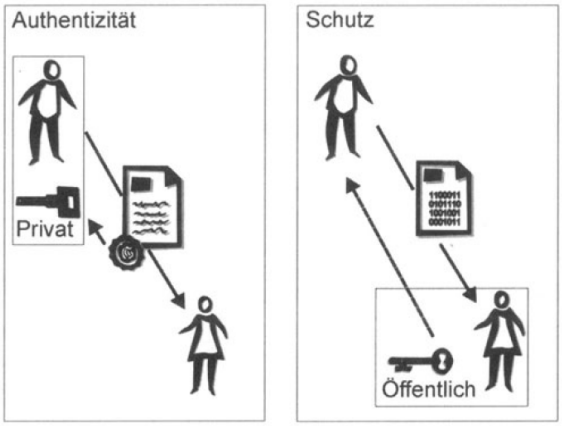


Abbildung 12 Die digitale Signatur dient der Authentizität durch Signieren, sowie dem Schutz der Daten durch Verschlüsseln (Bitzer & Brisch, 1999).

Das Prinzip der digitalen Signatur eines Dokuments besteht aus verschiedenen Schritten. Die Basis bildet das sogenannte *Public-Key-Verfahren*, welches ein Verfahren der asymmetrischen Kryptographie ist. Das Prinzip des *Public-Key-Verfahrens* ist die Existenz zweier Schlüssel, welche eine Abhängigkeit über einen mathematischen Algorithmus aufweisen (Bitzer & Brisch, 1999). Der öffentliche Schlüssel (*public key*) dient dem Sender der Daten zur Verschlüsselung der Daten (Bitzer & Brisch, 1999). Aus dem öffentlichen Schlüssel ist es unmöglich, den privaten Schlüssel abzuleiten. Der private Schlüssel (*private key*) ist nur dem Empfänger zugänglich und dient der Entschlüsselung der empfangenen Daten (Bitzer & Brisch, 1999). Die Daten können nur mit dem privaten Schlüssel des Empfängers dekodiert werden und sind somit vor der Einsicht Dritter geschützt (Bitzer & Brisch, 1999).

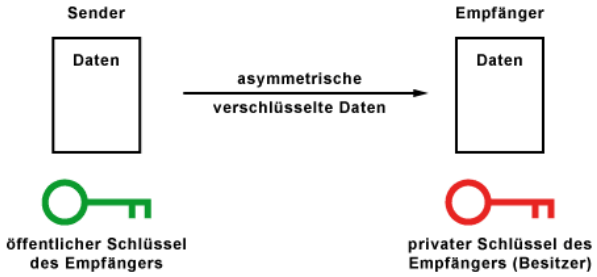


Abbildung 13 Die digitale Signatur dient der Authentizität durch Signieren, sowie dem Schutz der Daten durch Verschlüsseln (Elektronik-Kompendium.de, 2021).

Bei der digitalen Signatur von Dokumenten ist nicht der Empfänger der Inhaber des privaten Schlüssels, sondern der Dokumentenautor (Bitzer & Brisch, 1999). Für die Signatur von digitalen Dokumenten wird zuerst über eine Hash-Funktion ein Wert generiert, der den Inhalt des Dokuments absolut und eindeutig widerspiegelt (Bitzer & Brisch, 1999). Dieses Abbild des Dokumenteninhalts wird nun mit dem privaten Schlüssel des Autors enkodiert und mit dem original Dokument zusammengeführt (Bitzer & Brisch, 1999). Zusätzlich wird der Datei noch das sogenannte Zertifikat hinzugefügt, welches den öffentlichen Schlüssel des Dokumentenautors beinhaltet (Bitzer & Brisch, 1999). Dieses Zertifikat ist wiederum von einer vertrauenswürdigen Institution (CA (Certificate Authority)) mit deren privaten Schlüssel signiert.

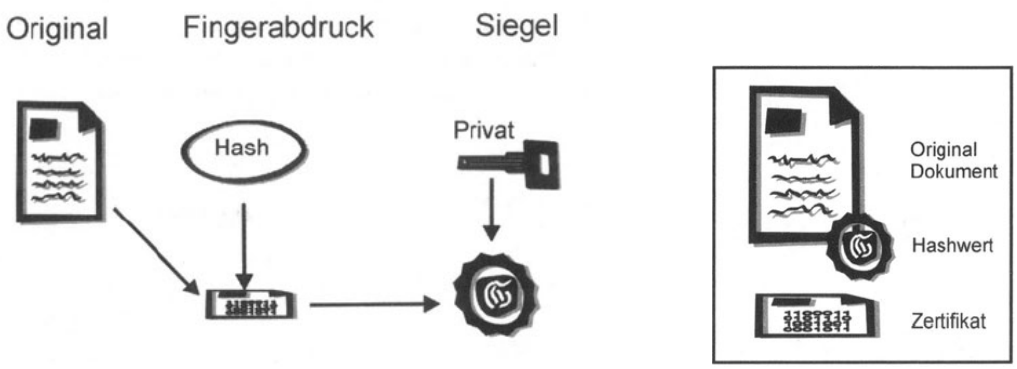


Abbildung 14 Ablauf der Signatur eines Dokuments (links) und resultierende Datei mit Originaldokument, Hash-Wert und Zertifikat (rechts) (Bitzer & Brisch, 1999).

Der Empfänger des Dokuments führt zunächst die Überprüfung der Echtheit und Vertrauenswürdigkeit des übermittelten Zertifikats mittels des öffentlichen Schlüssels der CA durch, um die Identität des Senders zu verifizieren. Mit dem im Zertifikat enthaltenen öffentlichen Schlüssel des Dokumentenautors wird dann die Signatur dekodiert (die Umkehrbarkeit der Entschlüsselung mittels des öffentlichen Schlüssels erfordert einen anderen Umgang mit den Variablen des RSA (Rivest–Shamir–Adleman)-Algorithmus) (Bitzer & Brisch, 1999; Wätjen, 2018; Hoffstein, Pipher & Silverman, 2014). Das Resultat ist der ursprüngliche Hash-Wert des originalen Dokuments (Bitzer & Brisch, 1999). Unter Anwendung derselben Hash-Funktion, welche von dem Dokumentenautor verwendet wurde, wird vom Empfänger erneut ein Hash-Wert des Dokuments berechnet (Bitzer & Brisch, 1999). Ist der erneut berechnete Has-Wert mit dem übermittelten Wert identisch, so wird bestätigt, dass das Dokument auf dem Übertragungsweg nicht verändert wurde (Bitzer & Brisch, 1999).

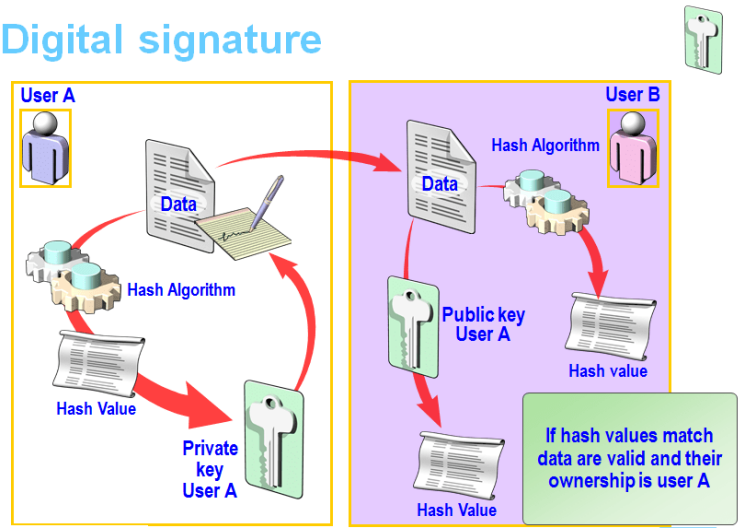


Abbildung 15 Validierung des empfangenen Dokuments anhand des Hash-Werts (Paganini, 2013).

Die europäische Signatur-Richtlinie sieht vor, dass elektronische Unterschriften, die rechtlichen Anforderungen (in Bezug auf Daten) in gleicher Weise erfüllen sollen, wie handschriftliche Unterschriften (Hühnlein & Korte, 2006). Nach BGB (Bürgerliches Gesetzbuch) §126 Absatz 3 kann *"die schriftliche Form [...] durch die elektronische Form ersetzt werden, wenn sich nicht aus dem Gesetz ein anderes ergibt"*. Außerdem können elektronisch signierte Dokumente in Gerichtsverfahren als Beweismittel zugelassen werden (Hühnlein & Korte, 2006). Die elektronische Signatur muss sicherstellen, dass diese dem Unterzeichner eindeutig zugeordnet wird (Identifikation) und nachgewiesen werden kann, wenn die Daten nachträglich geändert wurden (*Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*, 2014).

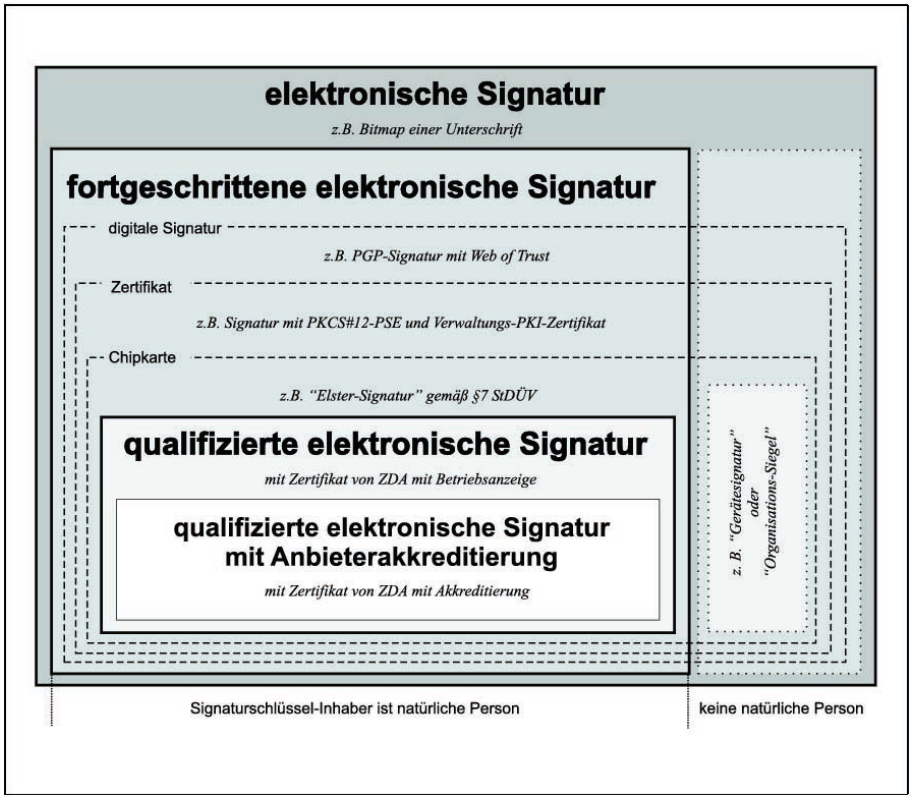


Abbildung 16 Abstufung verschiedener elektronischer Signaturen (Hühnlein & Korte, 2006).

Für die Signatur von XML-Dokumenten entwickelte eine Arbeitsgruppe des W3C das Signatur-Format XMLDSig (`https://www.w3.org/TR/xmldsig-core1/`) (Hühnlein & Korte, 2006; Yiu et al., 2015; Roessler et al., 2013). Mit XAdES wurde eine Erweiterung der XMLDSig-Spezifikation mit Konformität zu den europäischen Richtlinien zu elektronischen Signaturen geschaffen (Cruellas et al., 2003; *TS 101 903 - V1.3.2 - XML Advanced Electronic Signatures (XAdES)*, 2006). In Abbildung 17 werden verschiedene Arten der XML-Signatur dargestellt. Die Signatur kann demnach die zu signierenden Daten umschließen (*enveloping signature*), von den Daten getrennt sein (*detached signature*), oder in den Daten enthalten sein (*enveloped signature*) (Hühnlein & Korte, 2006).

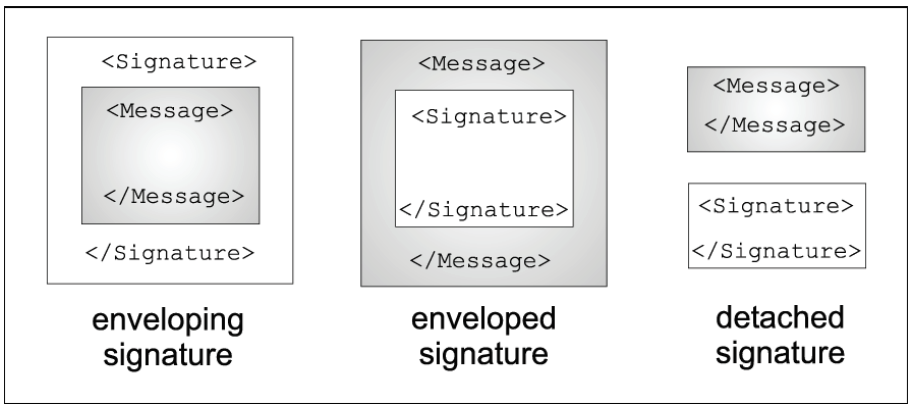


Abbildung 17 Unterschiedliche Realisierungen der XML-Signatur (Hühnlein & Korte, 2006).

Aus den vorigen Ausführungen geht hervor, dass der Dokumentenautor eines amtlichen Lageplans sich ein von einem akkreditierten Institut autorisiertes Zertifikat ausstellen lassen muss, dass dem Lageplan-Dokument hinzugefügt werden kann, und welches dem Empfänger bekannt ist.

Zu 4.: Vollständigkeit der Objektklassen

Für die zu modellierenden Objektklassen wird sich auf den vom BDVI bereitgestellten Objektkatalog in Unterabschnitt 3.1.1, sowie den Objektartenkatalog des AAA-Fachschemas und den im Datenmodell von XPlanung enthaltenen Klassen bzgl. des Bebauungsplans bezogen. Zusammen mit den Klassen, welche in CityGML, sowie existierenden ADEs (z.B. die UtilityNetwork ADE) bereits vorhanden sind, ist der Großteil der im urbanen Raum auftretenden Topologie und konzeptionellen Konstrukte abgedeckt.
Mit Bezug auf die Eingangsdaten bei der Erstellung eines Lageplans in Abbildung 8 ist in Tabelle 1 die Abdeckung der Eingangsdaten über Objekt-Klassen der verschiedenen Datenmodelle abgebildet. Diese Objekt-Klassen werden für der Entwicklung des Datenmodells des 3D-Lageplans übernommen.

Thematische Eingangsdaten	**Beispielobjekte**	**Abgedeckt durch Objektklassen von**
Planungsmodell	Gebäudemodell	CityGML
Liegenschaftskataster	Flurstück, Grundstücksgrenze, ...	AAA, CityGML, UtilityNetwork ADE
Bebauungsplan	Baulinie, Baugrenze, BMZ, ...	XPlanung
Bestandsvermessung	DGM, Baumbestand, Grenzneuvermessung, ...	AAA, CityGML
Zusätzliche Datensätze	Leitungspläne, ...	CityGML, UtilityNetwork ADE

Tabelle 1 Abdeckung der Eingangsdaten über Objektklassen verschiedener Datenmodelle, welche in das Datenmodell des 3D-Lageplans übernommen werden.

Nicht vernachlässigbar sind aktuelle und zukünftige städtebauliche Prozesse, Entwicklungen und Innovationen in der Technologie und Digitalisierung, die neue Objektarten mit sich bringen. Aktuelle Entwicklungen gehen in die Richtung von *Smart Cities*, alternative Formen der Mobilität (z.B. eMobility) und *Smart Buildings*, sowie Energiegewinnungssysteme für erneuerbare Energie in Gebäuden. In Tabelle 2 werden Objektklassen aufgeführt, welche neueste Entwicklungen mit sich bringen könnten und, ob diese für das Datenmodell des Lageplans von Bedeutung sind. Objekte, welche nur auf der technischen Ebene Auswirkungen haben, jedoch keinen topologischen und/oder rechtlichen Charakter bzgl. der räumlichen Domäne aufweisen, hat für den Lageplan, welcher raum-bezügliche Relationen zwischen Objekten darstellt, keine Relevanz.
Es ist festzustellen, dass sich im Bereich *Smart City*, *Smart Building* und autonomes Fahren die Innovationen in der Technologie selbst und weniger in topologischen Objekten zu finden sind. Die virtuelle Vernetzung der einzelnen Geräte, deren Kommunikation untereinander und die Entwicklung intelligenter Algorithmen für die Auswertung der Daten und effizientere Nutzung der Ressourcen stehen im Vordergrund (Jaana Remes & von der Tann, 2018; Morvaj, Lugaric & Krajcar, 2011; Erbstößer, 2019; *Strategie automatisiertes und vernetztes Fahren*, 2015).

Im Gegensatz dazu gibt es durch die Entwicklungen in der Mobilität zum Teil neue Objekte im Stadtbild. Dazu gehören bspw. Ladestationen für elektrische Fahrzeuge und *Mobility Stations*, welche als Knotenpunkt für intermodale Mobilität verschiedene Möglichkeiten an Fortbewegungsmittel bieten und eine Verkettung der unterschiedlichen Mobilitätsmedien ermöglichen (Garde, Jansen & Bläser, 2014; Laura Gebhardt & Wagner, 2018; Gernot Steinberg & Scherer, 2015; Carsten Sommer, 2016).
Auch im Bereich der erneuerbaren Energie werden innovative Energiesystem für Wohn- wie Industriegebäude aufkommen. Dies können bspw. Installationen wie Solarpanele, Wärmepumpen oder Biomassenboiler sein (*Innovation durch Forschung - Erneuerbare Energien und Energieeffizienz: Projekte und Ergebnisse der Forschungsförderung 2018*, 2019; Maarten De Groote, 2017; Maarten De Groote & Bean, 2017).
Für alle genannten Innovationen ist jedoch der Anschluss zu schnellem Internet erforderlich und damit der Breitbandausbau. Dieser Breitbandausbau und Anschluss an das Internet wird in den Planungen vermehrt eine wichtige Rolle einnehmen. Auch der Lageplan ist betroffen, indem bspw. Flächen von Grunddienstbarkeiten für die Verlegung von Kabeln, etc. in den Lageplan eingetragen werden müssen.

Innovation	Beispielobjekte	Relevanz für den Lageplan	Abgebildet durch Objektklasse von
Smart City	IoT (Internet of Things)-Infrastruktur/Breitbandausbau, Sensoren, ...	Nein	CityGML (z.B. Dynamizer), UtilityNetwork ADE
Smart Building	Smart meter, Gateways, Mikro-Kontroller, Breitbandausbau, Sensoren, ...	Nein	CityGML (z.B. Dynamizer), UtilityNetwork ADE
Autonomes Fahren	Digitale Infrastruktur, Sensoren, Breitbandausbau, ...	Nein	CityGML (z.B. Dynamizer), UtilityNetwork ADE
Mobilität	Mobility Stations (Car/Bike Sharing, ÖPNV (Öffentlicher Personennahverkehr)), Ladestationen, Breitbandausbau ...	Ja	CityGML (z.B. CityFurniture), UtilityNetwork ADE
Erneuerbare-Energie-Systeme	Photovoltaik-/Solarthermie-Anlagen, Wärmepumpen, Biomassenboiler, ...	Ja	CityGML (z.B. BuildingInstallation-Klasse), UtilityNetwork ADE

Tabelle 2 Innovationen und deren Relevanz für die Erstellung eines Lageplans.

Trends in der aktuellen Stadtplanung gehen hin zu einem umweltschonend mobilen, Lärmarmen, grünen, kompakten und heterogenen Stadtbild (*Die Stadt für Morgen: Umweltschonend mobil – lärmarm – grün – kompakt – durchmischt*, 2017; Umweltbundesamt, 2018). Dabei geht es u.a. um den Ausbau des ÖPNVs, die Reduktion des Individualverkehrs und die damit verbundene Nutzung der zurückgewonnen Flächen für Begegnungszentren und Grünflächen (*Die Stadt für Morgen: Umweltschonend mobil – lärmarm – grün – kompakt – durchmischt*, 2017; Umweltbundesamt, 2018). Kurze Wege und gute Vernetzung sollen das Verkehrsaufkommen minimieren. Eine Rolle spielen dabei auch die vorher genannten Konzepte der *Smart Cities*, *Smart Buildings* und der intermodalen innovativen Mobilität. In neuen Stadtkonzept werden dabei nur Vorgaben und Richtlinien gegeben, die dann von Innovationen implementiert werden. Konkrete mit einhergehende Stadtobjekte werden demzufolge nicht explizit genannt.

Grundsätzlich kann festgestellt werden, dass Innovationen gut auf die existierenden CityGML-Module abgebildet werden können. Dies ist möglich, da die Module sehr generell gehalten sind und es dadurch gelingt das gesamte thematische Spektrum von urbanen Räumen zu erfassen. Um eine Innovation abzubilden, muss zunächst eine generalisierte thematische Abbildung auf ein Modul stattfinden. Anhand der modularen Zuweisung wird anschließend versucht, das innovative Objekt mit einer Klasse des Moduls zu assoziieren. Anschließend kann über anpassbare Code-Listen die genaue Identität des Objektes definiert werden (Gröger et al., 2012; Kolbe et al., 2020).

Zu 5.: Ableitung eines 2D-Lageplans von einem 3D-Lageplan

Es stehen zwei Ansätze für die Generierung eines 2D-Lageplans aus einem 3D-Lageplan zur Verfügung:

1. Integration einer zusätzlichen simultanen 2D-Repräsentation der 3D-Objekte im Datenmodell
2. Projektion der 3D-Objekte auf eine 2D-Ebene

Zu *1.* bietet CityGML bereits das LOD-Konzept, welches erlaubt, simultan verschiedene geometrische Repräsentationen eines Objektes im Datensatz bereitzustellen (Gröger et al., 2012; Kolbe et al., 2020). Im Verarbeitungsschritt der Erstellung eines 2D-Lageplans aus einem vorhandenen 3D-Lageplan müsste dann nur noch die entsprechende Geometrie aus dem Datensatz extrahiert werden und könnte ohne weitere Prozessierung in einen 2D-Lageplan geschrieben werden. Dies kann über entsprechende Bibliotheken oder API (Application Program Interface)s über einfache XML-Anfragen geschehen.
Für *2.* muss für jedes geometrische Objekt des 3D-Lageplans eine geometrische Transformation bzw. Projektion nach 2D stattfinden, bevor das Objekt in den 2D-Lageplan geschrieben werden kann. Dies könnte bspw. in einer FME durchgeführt werden.
Da in CityGML von Grund auf die Möglichkeit besteht, für jedes topologisches Objekt mehrere geometrische Repräsentationen zu bereitzustellen, muss dies nicht in der Entwicklung des Datenmodells für den 3D-Lageplan berücksichtigt werden. Es besteht allerdings die Möglichkeit, über ein UML-Profil und OCL eine Bereitstellung von 2D-Geometrien für jedes Objekt zu erzwingen, wenn das Instanz-Dokument gegen das XML- bzw. Schematron-Schema validiert wird. Die Notwendigkeit einer integrierten 2D-Repräsentation wird nochmals in Punkt *6.* Bedeutung finden.

Zu 6.: Analysen und Berechnungen

Für die automatisierte und digitalisierte Prüfung des (amtlichen) Lageplans ist es notwendig, dass das Datenmodell des 3D-Lageplans darauf ausgelegt ist, dass Computerprogramme ohne große Bearbeitungsschritte Objektgruppen aus dem Datensatz extrahieren und aufgrund deren Geometrien topologische Analysen, sowie Berechnungen durchführen können. Zunächst muss betrachtet werden, welche Analysen und Berechnungen mit Grundlage des Lageplans zum Tragen kommen, wenn ein bspw. ein Bauvorhaben geprüft wird. Der Lageplan enthält neben der Topographie konzeptionelle Konstrukte, wie Grundstücksgrenzen, Grunddienstbarkeiten, Baulasten, Abstandsflächen und Regulierungen aus dem Bebauungsplan. Werden bspw. Festsetzungen aus dem Bebauungsplan missachtet (z.B. Grundstücksgrenzen werden nicht eingehalten), so ist das Bauvorhaben nicht zulässig (Noack et al., 2005). Auf Grundlage der im Lageplan enthaltenen Festsetzungen des Bebauungsplans, sowie des geplanten Bauwerks kann also z.B. überprüft werden, ob

- das geplante Bauwerk auf zulässigen bebaubaren Flächen errichtet wird,
- GRZ, GFZ und BMZ des geplanten Bauwerks mit den im Bebauungsplan festgesetzten Werten übereinstimmen,
- Baulinien und Baugrenzen eingehalten werden,
- die Art der baulichen Nutzung eingehalten wird,
- das Maß der baulichen Nutzung eingehalten wird,
- die Bauweise eingehalten wird und
- sonstige Festsetzungen im Bebauungsplan eingehalten werden.

Im Falle der Abstandsflächen muss der entsprechende geltende Gesetzestext in der BauO (Bauordnung) berücksichtigt werden. In BauO NRW §6 mit Stand vom 20.8.2020 werden an Abstandsflächen u.a. folgende Bedingungen gestellt:

- Abstandsflächen sind vor Außenwänden, welche an Grundstücksgrenzen errichtet werden, wenn nach planungsrechtlichen Vorschriften an die Grenze gebaut werden muss/darf, nicht notwendig
- Abstandsflächen müssen auf dem Grundstück selbst liegen bzw. dürfen sich auf ganz oder teilweise auf andere Grundstücke erstrecken, wenn öffentlich-rechtlich gesichert ist, dass sie nur mit in der Abstandsfläche zulässigen baulichen Anlagen überbaut werden
- Abstandsflächen können auf öffentlichen Verkehrs-, Grün- und Wasserflächen liegen, wenn sie sich nicht mehr als bis zu deren Mitte erstrecken
- Abstandsflächen dürfen sich nicht überdecken, außer bei

- Außenwänden, die in einem Winkel von mehr als 75 Grad zueinanderstehen,
- Außenwänden zu einem fremder Sicht entzogenen Gartenhof bei Wohngebäuden der Gebäudeklassen 1 und 2,
- Gebäuden und anderen bauliche Anlagen, die in den Abstandsflächen zulässig sind oder gestattet werden

Die Beispiele der Prüfung der Einhaltung von Festsetzungen aus dem Bebauungsplan, sowie die Validierung der Abstandsflächen erfordern, dass die Objekte sowohl auf topologische Relationen analysiert, als auch Flächen- und Volumenberechnungen durchgeführt werden können. So ist es bspw. für die Prüfung, ob das geplante Bauwerk innerhalb von bebaubaren Flächen liegt notwendig, die topologischen Relation (siehe Abbildung 18) des *Footprints* des Bauwerks in Bezug auf die bebaubaren Flächen zu bestimmen.
Für die Berechnung der GFZ sind auch die Flächen der Geschosse innerhalb des Gebäudes notwendig, d.h. für das Gebäudemodell ist es nicht ausreichend, die äußerer Hülle bereitzustellen. Es muss zusätzlich mindestens ein Grundriss jedes der Geschosse im Gebäudemodel enthalten sein.
Im Gegensatz zu Prüfungen von Bebauungsplan-Festsetzungen, die eine Flächenberechnungen in 2D erfordern (z.B. GRZ), sind auch volumenbasierte Rechnungen notwendig (z.B. BMZ).
Aus der Notwendigkeit von 2D-Analysen bei der digitalisierten (und automatisierten) Prüfung folgt, dass neben der 3D-Darstellung der Objekte im 3D-Lageplan trotzdem eine 2D-Repräsentation enthalten sein sollte. Die Prüfsoftware würde dann anhand bereitgestellter digitalisierter Bebauungspläne und für das Baugebiet geltenden öffentlich- und privat-rechtliche Regeln das Bauvorhaben anhand des eingereichten Lageplans prüfen. Die Prüfsoftware muss sich auf einen Standard verlassen können, den der eingereichte Lageplan erfüllt. Daher wäre es sinnvoll, über ein UML-Profil eine verpflichtende Bereitstellung von 2D-Geometrien für Lageplan-Objekte zu fordern. Die Validität des Lageplan-Dokuments kann vor der Prüfung über eine XSD- und/oder Schematron-Validierung sichergestellt werden.

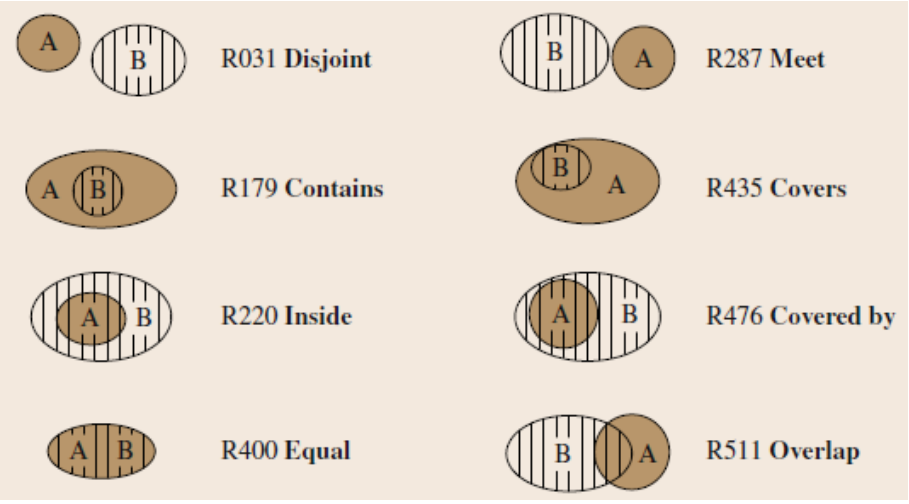


Abbildung 18 Verschiedene topologische Relationen in 2D (Gröger & George, 2012).

Zu 7.: Berücksichtigung von Anwendungsfeldern-/fällen

Wie in Abschnitt 2.2 beschrieben, können die Domänen der Anwendbarkeit des Lageplans in drei Bereiche aufgeteilt werden:

- Entwurfsverfasser*in
- Bauantragssteller*in
- Baugenehmigungsverfahren

Zusätzlich werden in Noack et al. (2005) weitere Anwendungsfälle genannt, für welche der (amtliche) Lageplan Basisdaten liefert:

- Gebäudeeinmessung
- Grob- und Feinabsteckung
- Baubegleitende Absteckung
- Absteckung von Außenanlagen
- Einmessungsbescheinigung
- Beleg für Banken (für Baufinanzierung), Versicherungen, Notar, Gutachter, Nachbarbeteiligung
- Überwachungs- und Baukontrollmessungen

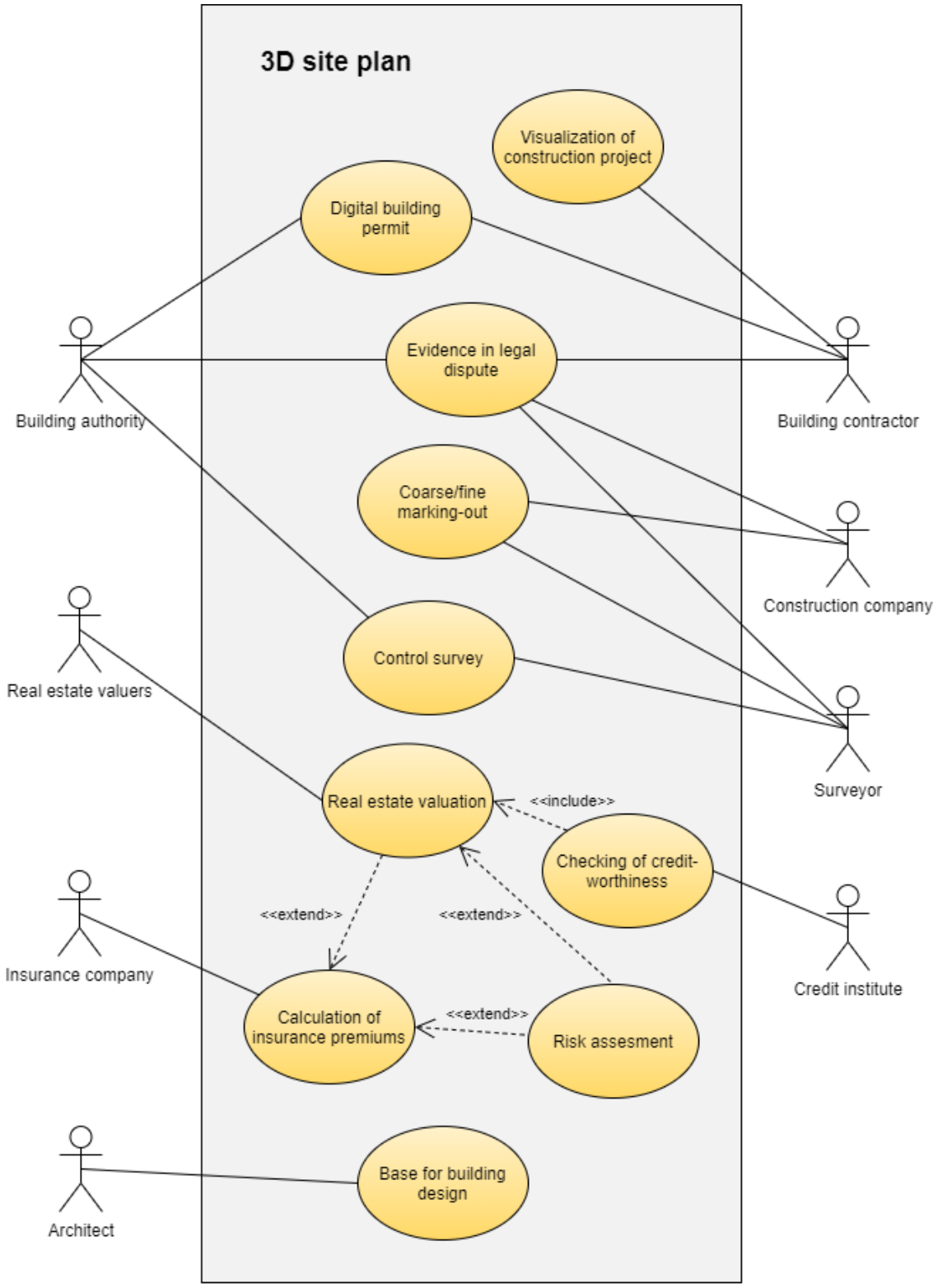


Abbildung 19 Use Case Diagramm für den Anwendungsfall des Lageplans.

Anwendungsfeld	**Anwendungsfall**	**Anforderung an das Datenmodell**
AEC-Software/Design/Gebäudemodell	Der Lageplan als Basis des Gebäudeentwurfs im Kontext der rechtlichen und topographischen Begebenheiten.	Das Datenmodell bzw. enthaltene Objekte müssen mit Software-Konzepten aus dem Planungssektor kompatibel sein.
GIS-Software/Planung	Digitale Prüfung/Erstellung des Lageplans (über 2D/3D-Operatoren).	Das Datenmodell muss die logische Aggregation von thematischen Objekten in Ebenen (*Layer*) unterstützen.
Digitaler Bauantrag	Digitales Einreichen der Bauvorlagen insbesondere des Lageplans.	Der Lageplan muss dem vom Softwaresystem der Baubehörde definierten Spezifikation des Datenmodells, sowie des Datenformats entsprechen.
Digitales/automatisiertes Baugenehmigungsverfahren	Digitale/automatisierte Prüfung des Lageplans bzgl. der Zulässigkeit des Bauvorhabens.	Das Datenmodell muss dem von der Prüfsoftware definierten Spezifikationen entsprechen.
Gebäudeeinmessung	Aktualisierung des Gebäudemodells im ”as-built”-Zustand und Einpflegung in amtliche Datenbank/Liegenschaftskataster.	Das Datenmodell muss zusätzliche Attribute für ”as-planned”-Koordinaten und ”as-built”-Koordinaten für geplante Objekte bereitstellen.
Grob- und Feinabsteckung	Dem Lageplan werden Koordinaten des geplanten Bauwerks für die Grob- und Feinabsteckung vor Baubeginn entnommen.	Möglichkeit der Extraktion von Koordinatenlisten von bestimmten Objekten muss gegeben sein.
Einmessungsbescheinigung	Der Lageplan dient als Referenz beim Nachweis der Einhaltung genehmigter Grundfläche und Höhenlage nach Baubeginn.	2D-/3D-Analysen müssen vom Datenmodell unterstützt werden.
Baukontrollmessung	Der Lageplan dient als Referenz beim Nachweis der Einhaltung Baugenehmigung während dem Baubetrieb.	2D-/3D-Analysen müssen vom Datenmodell unterstützt werden.
Beleg für Banken, Versicherungen, Gutachter, ...	Der Lageplan als Hilfe bei der Grundstückswertermittlung, Risikobewertung, etc..	2D-/3D-Analysen, sowie Wertermittlung müssen vom Datenmodell unterstützt werden.

Tabelle 3 Anwendungsfelder des (amtlichen) Lageplans und deren Anforderungen an das 3D-Lageplan-Datenmodell.

AEC-Software/Design/Gebäudemodell: Trotz Versuche der Angleichung der Modellierung ist die Kompatibilität von GIS-Datenmodellen zu Software aus dem Bereich des AEC schwierig zu gestalten, da die Modellierungsparadigmen stark verschieden sind (Kutzner et al., 2020; Ohori et al., 2018; Kaden & Kolbe, 2016). Daher ist es sinnvoller, dass die entsprechenden Software Schnittstellen bereitstellen und die CityGML-Datensätze ”on-the-fly” in virtuelle Objekt-Klassen im Zieldatenmodell transformiert. Ein verlustfreie Transformation erfordert allerdings eine Angleichung zwischen den Datenmodellen der beiden Bereichen.

GIS-Software/Planung: Um den digitalen Lageplan in klassischer GIS-Software zur digitalen Planung und anschließender Prüfung verwenden zu können, muss das Datenmodell es ermöglichen, bei der Erstellung des Lageplans flexibel oder auch vordefiniert thematische Aggregationen von Objekten zu sogenannten *Layer* bereitzustellen. Dies kann über der CityGML-Klasse *CityObjectGroups* geschehen (Gröger et al., 2012; Kolbe et al., 2020).

Digitaler Bauantrag/digitalisiertes/automatisiertes Baugenehmigungsverfahren: Da zum Zeitpunkt der Entstehung der Arbeit, außer dem GML-Format, noch keine Bedingungen an das Datenmodell/Datenformat des Lageplans bzgl. der Prüfsoftware gestellt wurden, können hier noch keine weiteren Spezifikationen der Baubehörden in die Entwicklung des Datenmodells einfließen (Theiler, 2020a).

Gebäudeeinmessung: Nach Fertigstellung des Gebäudes ist es laut VermKatG (Vermessungs- und Katastergesetz) NRW §16 Absatz 2 verpflichtend, eine Gebäudeeinmessung durchzuführen. Dabei wird das Gebäude ”as-built” von einem*einer Vermessungsingenieur*in aufgenommen. Anschließend kann das Gebäudemodell mit den realen Koordinaten aktualisiert werden und eine amtliche Datenbank für Stadtmodelle eingepflegt werden. Die Bereitstellung verschiedener Versionen des Bauwerks kann in CityGML über das *Versioning*-Modul realisiert werden, welches die Bereitstellung verschiedener Versionen von CityGML-Objekten erlaubt (Kolbe et al., 2020; Chaturvedi, Smyth, Gesquière, Kutzner & Kolbe, 2016; Kutzner et al., 2020).

Fein- und Grobabsteckung: Vor Baubeginn wird durch die Extraktion der Gebäudekoordinaten eine Fein- bzw. Grobasbteckung vorgenommen. Die Koordinatenliste des jeweiligen CityGML-Objekts sind von Grund auf in den Objektinstanzen enthalten. Im Gegensatz zur Grobabsteckung, die die Lage der Baugrube festlegt, handelt es sich bei der Feinabsteckung um die Übertragung der Gebäudeeckpunkte in das Baufeld (Schröder, 2018a).

Einmessungsbescheinigung/Baukontrollmessung: Sinn der Einmessungsbescheinigung bzw. der Baukontrollmessungen sind die Überprüfung der Einhaltung der genehmigten Sollwerte des Bauvorhabens (Schröder, 2018b). Hierfür werden die von einem/r Vermessungsingenieur/in aufgenommenen Koordinaten vor Ort mit den Koordinaten des Planungsmodells im (amtlichen) Lageplan verglichen und auf Abweichung überprüft. Wie auch schon im Paragraph zur *Gebäudeinmessung* können die örtlichen Koordinaten des Bauwerks, die während des Bauablaufs erhoben werden, als Versionsattribut in das jeweilige CityGML-Objekt integriert werden. Über Internetzugriff auf das Portal der Baubehörde könnte der*die Vermessungsingenieur*in nach jeder Kontrollmessung die Bauwerkskoordinaten zeitnah aktualisieren und sogleich der Baubehörde für die Prüfung zugänglich machen. Damit würde der Lebenszyklus des Bauwerks von Planung bis Fertigstellung bzgl. der Baukontrolle dynamisch erfasst werden.

Basis für Belege für Banken, Versicherungen, etc.: (Amtliche) Lagepläne finden auch in anderen Sektoren als dem Bausektor ihre Anwendung. Bspw. kann der Lageplan Versicherungen bei der Risikobewertung für Berechnung von Beiträgen hilfreich sein, indem die Sicherungsbeschreibung und -prüfung auf Basis des Lageplans geschieht. Z.B. können auf Grundlage des 3D-Lageplans bspw. Hochwassersimulationen durchgeführt werden (Simon, 2012; Trometer, Schilling, Heyer & Mager, 2017; Kilsedar, Fissore, Pirotti & Brovelli, 2019). Denkbar sind auch Lawinen- oder Hangrutschsimulationen, Explosionssimulationen (wie in Willenborg (2015)), Brandsimulationen, etc.. Insbesondere für die Schadensberechnung bzw. Wertermittlung sollte das Lageplan-Datenmodell demnach Information über verwendete Baumaterialien und Bodenrichtwerte beinhalten können.
Bei der Gutachtenerstellung für die Ermittlung von Gebäudeversicherungssummen gehen Bauart, Gebäudeklasse, sowie Grundfläche und Rauminhalt mit in die Berechnung der Wertermittlung ein (Continental Sachversicherung AG, 2005). Für Grundfläche und Rauminhalt werden Flächenberechnungen in 2D bzw. Volumenberechnungen in 3D notwendig. Wie schon zuvor impliziert dies jeweils eine 2D- und 3D-Repräsentation der Lageplan-Objekte. Auch bei den Unterlagen für eine Baufinanzierung werden Lagepläne gewünscht, um die Kreditwürdigkeit zu prüfen und eine Immobilienbewertung durchführen zu können (Kreissparkasse Esslingen-Nürtingen, 2016; Volksbank am Würtemberg eG, 2016; R+V Versicherungen, 2015).

Zusammenfassung der Anforderungen an das 3D-Lageplan-Datenmodell

- 3D-Repräsentation aller Lageplan-Objekte (inklusive des Geländes) für Bauvorhaben-bezogene Berechnungen, sowie Simulationen
- Kompatibilität zu existierenden Standards (LADM, INSPIRE, AAA, XPlanung)
- Elektronische Signatur
- Metadaten über im Lageplan dargestellte Daten
- Referenz zu Beteiligten im Bauvorhaben
- Innovationen werden auf existierende CityGML-Klassen abgebildet und über Codelisten spezifiziert
- Zusätzliche Bereitstellung einer 2D-Repräsentation für jedes Lageplan-Objekt
- Thematische Aggregation der Lageplan-Objekte zu *Layer*
- Verschiedene Versionen des Bauvorhabens von Planung, über Beginn des Bauzyklus, bis zur Fertigstellung des Bauwerks
- Baumaterialien und Bodenrichtwerte für Wert- und Schadensermittlung

3.1.4 Thematische Aufteilung der Lageplan-Objekte

Klassifizierung der Lageplan-Objekte

Wird die Objektmenge des Lageplans betrachtet, ist festzustellen, dass dieser aus zwei Arten von Objekten besteht. Zum einen aus Objekten, die sichtbar, greifbar und über Messtechnik und Sensoren erfassbar sind und die Umwelt und die Erdoberfläche charakterisieren und formen. Zum anderen sind es Objekte, welche von der Menschheit auf konzeptioneller Ebene konstruiert wurden, um (rechtliche und soziale) Ordnung in der Welt zu etablieren.
Folglich lassen sich erst genannte Objekte als Topographie bezeichnen, letztere als konzeptionelle Domäne oder konzeptionelle Konstrukte.
Des weiteren enthält der Lageplan Objekte, welche noch nicht konzeptionell oder topographisch existent, sondern zukünftig geplant sind. So lassen sich die Objekte weiter unterteilen, nämlich in topographische und konzeptionelle Objekte, welche existent oder geplant sind.
Der Unterschied zwischen konzeptionellen Konstrukten und topographischen Objekten wird als signifikanter angesehen, da die Existenz einen Status bzw. Zustand eines Objektes beschreibt und nicht dessen Charakter an sich. Die generelle Klassifizierung der Lageplan-Objekte sieht deshalb zwei Kategorien vor: **Konzeptionelle Konstrukte** und **Topographie**
Der Zustand eines Objektes kann durch Bereitstellung eines entsprechenden Attributs abgebildet werden.

Die für den Lageplan relevanten thematischen Bereiche der Topographie werden von CityGML standardmäßig bedient. Für die Modellierung der Versorgungsleitungen steht bereits die Erweiterung *UtilityNetworkADE* zur Verfügung. Die konzeptionelle Domäne wird von CityGML jedoch nicht ausreichend abgedeckt, sodass v.a. dieser Bereich im Augenmerk der Entwicklung der Lageplan ADE (SiteplanADE) liegt.

Topographie	**Konzeptionelle Konstrukte**
• Bauwerke • Gebäude • Gelände • Vegetation • Landnutzung • Wasserkörper • Transportflächen • Versorgungsleitungen	• Landadministration/Kataster • Landplanung • Baurecht • Konflikte

Tabelle 4 Thematische Aufteilung der Objektmenge im Kontext des Lageplans.

Weitere thematische Einteilungen der Lageplan-Objekte

Um weitere thematische Zusammenhänge zugänglich zu machen wird von der CityGML-Klasse *CityObjectGroup* Gebrauch gemacht. Diese erlaubt die Zusammenfassung verschiedener Objekte zu logischen und thematischen Aggregationen (Gröger et al., 2012; Kolbe et al., 2020). Diese Aggregationen können analog als Ebenen in GIS-Software-Programmen aufgefasst werden, anhand derer verschiedene Analysen durchgeführt werden können, oder auch die Datenzugehörigkeit nachvollzogen werden kann. In Tabelle 5 sind verschiedene thematischen Gruppierungen dargestellt. Weitere *CityObjectGroups* können auch durch Bereitstellen individueller Code-Listen angepasst werden.

Thematische Aggregation	**Anwendung**
Spatial plan	• Nachvollziehbarkeit der ursprünglichen Objektzugehörigkeit • Ableitung des Bebauungsplans im Bereich des Lageplans
Cadastral map	• Nachvollziehbarkeit der ursprünglichen Objektzugehörigkeit • Ableitung eines aktualisierten Katasterauszugs im Bereich des Lageplans mit Bauvorhaben
Existing objects	• Übersicht über aktuellen Bestand des Baugebiets bzgl. Topographie und konzeptioneller Objekte • Klare Unterscheidung zwischen geplanten Objekten und aktuellem Bestand • Planungsgrundlage für Entwurfsverfasser*in
Planned objects	• Übersicht über geplante Objekte im Baugebiet bzgl. Topographie und konzeptioneller Objekte • Klare Unterscheidung zwischen geplanten Objekten und aktuellem Bestand • Überlagerung mit existierenden topographischen und/oder konzeptionellen Objekten macht Konflikte und Interferenzen deutlich

Existing topographic objects	• Überlagerung mit geplanten topographischen und/oder existierenden/geplanten konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar
Planned topographic objects	• Überlagerung mit existierenden topographischen und/oder existierenden/geplanten konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar
Existing conceptual objects	• Überlagerung mit existierenden und/oder geplanten topographischen und geplanten konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar
Planned conceptual objects	• Überlagerung mit existierenden und/oder geplanten topographischen und existierenden konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar
Existing public law objects	• Überlagerung mit geplanten topographischen Objekten macht Konflikte und Interferenzen erkennbar
Existing private law objects	• Überlagerung mit geplanten topographischen Objekten macht Konflikte und Interferenzen erkennbar

Planned public law objects	• Überlagerung mit geplanten topographischen und geplanten konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar • Überlagerung mit privaten und öffentlichen Raum kann Aufschluss über Validität des Vorhabens geben
Planned private law objects	• Überlagerung mit geplanten topographischen und geplanten konzeptionellen Objekten macht Konflikte und Interferenzen erkennbar • Überlagerung mit privaten und öffentlichen Raum kann Aufschluss über Validität des Vorhabens geben
Public space objects	• Evaluierung von bspw. Abstandsflächen und anderen relevanten Objekten in Bezug zum öffentlichem Raum
Private space objects	• Evaluierung von bspw. Abstandsflächen und anderen relevanten Objekten in Bezug zu privaten Objekten und privaten Raum

Tabelle 5 Vorschläge thematischer Gruppierungen in *CityObjectGroups* und deren Anwendung.

3.1.5 Geometrie der Lageplan-Objekte in verschiedenen Lageplan-Versionen

Um den Lageplan in neben einer Version in 3D auch in 2D bereitzustellen ist es notwendig, die Geometrie für die verschiedenen Lageplan-Objekte für die jeweilige Version festzulegen. Grundsätzlich gilt, dass in CityGML 3.0 sogenannte *Spaces* in LOD0 als Punkt, Multi-Kurven oder Multi-Flächen dargestellt werden können, in LOD1 als Solid, in LOD2 als Solid, Multi-Flächen oder Multi-Kurven und in LOD3 als Solid, Multi-Kurven oder Multi-Flächen (Kolbe et al., 2020).

Flächenobjekte haben in jeglichem LOD eine Multi-Flächen-Repräsentation, jedoch nur in LOD0 zusätzlich die Möglichkeit, über Multi-Kurven dargestellt zu werden (Kolbe et al., 2020).

Für die Repräsentation eines Lageplans in 2D wird die Objekt-Geometrie auf die Geometrie Multi-Punkt, Multi-Fläche und Multi-Kurve beschränkt. Die Geometrien sollen planar sein (keine dreidimensionale Ausdehnung besitzen) und auf einer Referenzebene schweben.

In der 3D-Version hingegen sind 3D-Geometrien gewünscht. Bspw. sollen Flächen der Landnutzungsart, Verkehrsflächen, etc. dem Geländeverlauf folgen, Vegetationsobjekte wie Bäume oder Stadtbildobjekte wie Laternen in 3D dargestellt werden, generell alle Objekte so detail-getreu wie notwendig und möglich mit 3D-Geometrie repräsentiert werden.

Konzeptionelle Objekte aus dem Bereich Landplanung, Landadministration und Baurecht werden im 2D-Lageplan, analog zu den topographischen Objekten, mit den Geometrien Multi-Punkt, Multi-Fläche und Multi-Kurve dargestellt.

Im 3D-Lageplan steht es offen, die konzeptionelle Objekte auch mit volumetrischen Geometrien zu instanziieren, um zukünftige Analysen auf Konflikte, welche räumliche Ausdehnung berücksichtigen, möglich zu machen (siehe (Emamgholian et al., 2020)). Wie auch topographische Objekte im 3D-Lageplan, sollen flächen- und linienhafte Objekte der konzeptionellen Domäne dem Geländeverlauf folgen und über das Geländemodell drapiert werden.

Object type		**2D site plan geometry representation**		**3D site plan geometry representation**	
		Geometry type(s)	*LOD*	*Geometry type(s)*	*LOD*
Topographic objects	Water body	MultiSurface, MultiCurve	0-3	Solid, MultiSurface	1-3
	Vegetation	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, ImplicitGeometry	0-3
	Transportation	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface	0-3
	City furniture	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, ImplicitGeometry	0-3
	Building	MultiSurface	0-3	Solid, MultiSurface	0-3
	Tunnel	MultiSurface, MultiCurve	0-3	Solid, MultiSurface	0-3
	Land use	MultiSurface	0-3	MultiSurface	0-3
	Utility network	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, MultiCurve, ImplicitGeometry	1-3
	Relief	MultiCurve	-	* (all)	-
Conceptional domain	Construction law	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, MultiCurve, Point	0-3
	Land use planning	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, MultiCurve	0-3
	Land administration	Point, MultiSurface, MultiCurve	0-3	Solid, MultiSurface, MultiCurve, Point	0-3

Tabelle 6 Vorgesehene Geometrietypen für die verschiedenen Lageplan-Objekte.

3.2 Entwicklung des 3D-Lageplan-Modells: SiteplanADE

3.2.1 Generelles Konzept für die Modellierung der Lageplan-Objekte

Für die Modellierung der topographischen Objekte stehen die im CityGML 3.0-Standard enthaltenen Klassen der verschieden thematischen Module bereit, die je nach Bedarf um notwendige Attribute (neuer ADE-Mechanismus) erweitert werden. Im Gegensatz dazu existieren für konzeptionelle Objekte aus Landplanung, Landadministration und Baurecht noch keine vordefinierten Klassen. Um die konzeptionelle Domäne abzubilden, gibt es zwei Möglichkeiten.

Die erste Option besteht daraus, wo möglich, die vorhandenen CityGML-Klassen mit den für die konzeptionellen Objekte erforderlichen Attribute mittels des neuen ADE-Mechanismus zu erweitern. Dies würde die Zweckbestimmung von bspw. Wasser- und/oder Straßenflächen aus dem Bebauungsplan betreffen, die ja im topographischen Sinne schon im CityGML-Standard enthalten sind, jedoch noch keine Attribute aus dem Kontext des Bebauungsplan besitzen. Die fehlenden Attribute würden dann über den ADE-Mechanismus bereitgestellt werden. Konzeptionelle Klassen, welche nicht auf bereits vorhandene CityGML-Klassen abgebildet werden können, müssen von einer *Core*-Modul-Klasse abgeleitet werden. Vorteil dieses Ansatz ist, dass schon auf vorhandene Klassen des CityGML-Standards zurückgegriffen werden kann. Zu beachten ist allerdings, dass der CityGML-Standard auf die topographische Abbildung der Realität fokussiert ist (Kolbe, 2019). D.h., die CityGML-Klassen der topographischen Module, welche auch Bezug zur konzeptionellen Domäne aufweisen, leiten sich von der CityGML-*Core*-Klasse *AbstractPhysicalSpace* ab. Konzeptionelle Objekte sind jedoch dem logischen bzw. konzeptionellen Raum der *AbstractLogicalSpace* zuzuordnen (Kutzner et al., 2020). Sie weisen nämlich keinen physikalischen Charakter auf, haben jedoch konzeptionelle Eigenschaften, die die Realität betreffen.

Ein weiterer Nachteil dieses Ansatzes ist, dass die Kompatibilität und Interoperabilität zu internationalen Standards erschwert wird. Da es bspw. definierte Abbildungsregeln zwischen XPlanung und INSPIRE (siehe (Duan & Benner, 2019)) oder ALKIS und LADM (siehe (Seifert, 2012)) gibt, kann das in Unterabschnitt 3.1.3 unter Punkt 2. beschriebene Konzept der Modellierung in den verschiedenen Ebenen von international zu national nicht angewandt werden, da alles auf der internationalen CityGML-Ebene modelliert werden würde.

Die zweite Option, sieht es vor, die konzeptionellen Klassen als eigenständige Klassen vom CityGML-*Core*-Modul abzuleiten. Die Klassen der konzeptionellen Domäne werden also von einer existierenden CityGML-Klasse des *Core*-Moduls neu abgeleitet. Hierfür wird die Klasse *AbstractLogicalSpace* als Superklasse dienen. Laut der Spezifikation von CityGML 3.0 stehen die logischen Volumenkörper für nicht (zwangsläufig) physikalische Einheiten der Welt mit thematischer Bedeutung oder Aggregation anderer physikalischer Objekte (Kolbe et al., 2020; Kutzner et al., 2020). Daher werden die Klassen der konzeptionellen Konstrukte Subklassen von der CityGML-Klasse *AbstractLogicalSpace* sein. Die neue Definition von Klassen erlaubt es, die Ebenen-Modellierung aus Unterabschnitt 3.1.3 unter Punkt 2. zu realisieren. Dadurch lassen sich Datensätze z.B. aus dem INSPIRE oder XPlanung-Standard nahezu 1:1 auf die Objektklassen des Lageplan-Modells abbilden.

Ein weiterer Vorteil der zweiten Option bezieht sich auf die Arbeitsschritte bei der Erstellung eines Lageplans nach dem SiteplanADE-Modell im CityGML-Format. Es wird angenommen, dass die Eingangsdaten für die Erstellung eines 3D-Lageplans nach dem SiteplanADE-Datenmodell zur Prä-Prozessierung in eine FME gelesen werden. Dort liegen die eingelesenen Daten als Format-neutrale Objekte mit Attributen in FME-interner Repräsentation vor. Auch für jede Ausgangsdatei sind Format-neutrale Ziel-Objekte mit entsprechenden Attributen vorhanden, auf welche die Eingangsdaten-Objekte abgebildet werden sollen, um diese dann schlussendlich in das gewünschte Dateiformat zu schreiben.
Werden Eingangsdaten also Objekte mit identischer oder ähnlicher FME-internen Repräsentation (bezieht sich auf Attribut-Namen und Datentyp der Attribute) eingelesen, wie die Ziel-Objekte der Ausgangsdatei, so verringern sich die Verarbeitungsschritte innerhalb der FME immens, um die Attribute der Eingangsobjekte auf die der Ausgangsobjekte anzupassen. Im besten Fall können die Attribute der Eingangsdaten-Objekte sogar direkt ohne Transformation auf das Ziel-Objekte abgebildet werden und diese dann direkt in die Ausgangsdatei im Ausgangsformat geschrieben werden. In Abbildung 20 ist sinngetreu dargestellt, welche Vorteile kompatible Standards bei dem Schreiben der Eingangsobjekte in das Zielformat mit sich bringt. Auf der linken Seite ist das Eingangsdaten-Objekt-Template in interner Darstellung visualisiert. Auf der rechten Seite das Ziel-Objekt-Template, welches dann in die Zieldatei im Zieldatenformat geschrieben wird. Attributname und -datentyp der internen Objekte werden jeweils von Eingangs- und Ausgangsdatensatz übernommen. Stimmen die Attribute mit Namen und Datentyp überein, so kann das Eingangsobjekt direkt auf das Ausgangsobjekt abgebildet werden, da beide Objekte ja in der FME-internen Repräsentation vorliegen. Das Ausgangsobjekt kann dann über einen *Writer* in das Zieldatenformat transformiert und in eine Datei geschrieben werden.

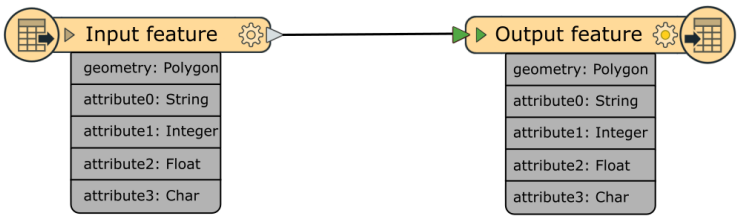


Abbildung 20 Schematische Darstellung des einfachen Schreibens in ein neues Datenformat bei Kompatibilität des Eingangs- und Ausgangsdatenmodell. Die Abbildung ist sinngemäß zu verstehen und hat die Reader/Writer-Funktionalität der Safe Software FME zur Vorlage.

Die vorangegangene Beschreibung der Vereinfachung der Arbeitsschritte durch das Wegfallen der Datentransformation wird durch die Kompatibilität in der Modellierung der SiteplanADE erreicht. Es wird deshalb versucht werden, sich bei der Modellierung der konzeptionellen Klassen der SiteplanADE so nah wie möglich an den existierenden Standards LADM, INSPIRE, XPlanung und AAA zu halten.

Im Folgenden werden die verschiedenen Pakete der SiteplanADE inhaltlich erläutert und Vorschläge zu deren Anwendung angeführt. Für die genaue Spezifikation der Klassen wird auf Anhang C verwiesen.

3.2.2 Package: Core

Inhaltliche Beschreibung

Bereits auf höchster CityGML-Modellierungsebene werden Objekte für den Anwendungsfall des Lageplans erweitert. Dafür werden den vorhandenen Klassen zusätzliche Attribute mittels des neuen ADE-Mechanismus (Erweiterung der ADE-Klasse) bereitgestellt.
Die Erweiterungen dienen einerseits der Bereitstellung von Metadaten für die im Lageplan enthaltenen Objekte bzgl. ihres rechtlichen Charakters, der Datenqualität, sowie der Herkunft und sollen für Transparenz und Rechtssicherheit sorgen. Dem Lageplan-Wurzel-Element (*CityModel*) können Attribute bereitgestellt werden, welche sich auf die am Lageplan beteiligten Person beziehen.
Des weiteren werden für die 3D- bzw. 2D-Darstellung des Lageplans Attribute für jedes Objekt bereitgestellt, die den Namen/Attribut-Bezeichner der Geometrie für die jeweilige Darstellung beinhalten.
Auch der Objekt-Status kann über einen Enumerations-Wert in ”existierend”, ”geplant” oder ”wegfallend” spezifiziert werden.
Zusätzlich kann jedem Objekt eine verbesserte Geometrie hinzugefügt werden. Dies eignet sich bei Neuvermessungen von Objekten dritter, die zu ungenaue Geometrie aufweisen. Durch die Speicherung der Geometrie im Attribut der verbesserten Geometrie, kann sowohl die ursprüngliche Geometrie, als auch die neue Geometrie im entsprechenden Objekt enthalten sein.

Mögliche Anwendung

Vor allem bei nachrichtlich übernommen Objekten im Lageplan (wie z.B. Versorgungsleitungen) ist es sinnvoll, deren Herkunft kenntlich zu machen, um zu verdeutlichen, dass der*die Verfasser*in des Lageplans für die Richtigkeit dieser Daten nicht rechtlich haftbar gemacht werden kann.
Auch sollten die für die Anfertigung des Lageplans verantwortliche Personen in den Daten hinterlegt werden um Transparenz zu schaffen.

3.2.3 Package: Building

Inhaltliche Beschreibung

Die Gebäudeklassen des CityGML-Standards werden nur durch wenige Attribute erweitert, welche v.a. für die Automatisierung der Abstandsflächenberechnung notwendig sind oder der Prüfung der Zulässigkeit des Gebäudes im jeweiligen Baugebiet dienen.
Insbesondere werden Attribute deshalb definiert, um die Art und Funktion der thematischen Begrenzungsflächen näher zu beschreiben. Dies ist v.a. für Abstandsflächenberechnungsalgorithmen relevant, da diese je nach Gesetzesvorlage Entscheidungen für oder gegen eine Notwendigkeit von Abstandsflächen häufig anhand des Charakters und Funktionalität der Wand im Kontext des Gesamteindrucks der Gebäudekonfiguration treffen. Beispiele für die Art und Funktion wären bspw. Giebel, Fassade, Seitenwand einer Gebäudeinstallation, o.ä..

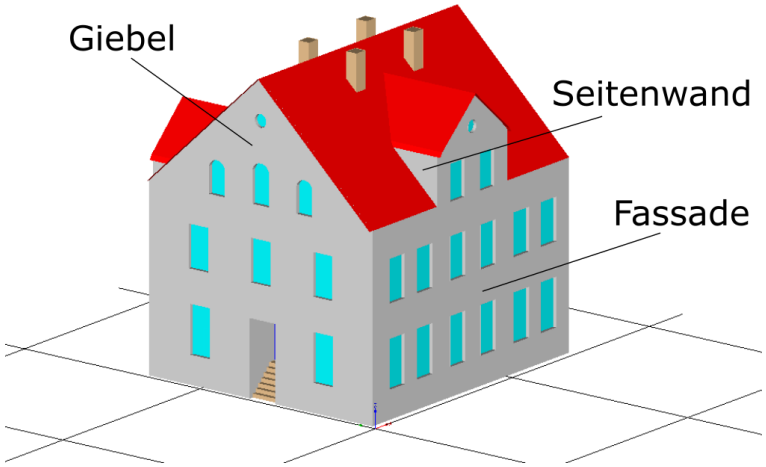


Abbildung 21 Darstellung von möglichen näheren Spezifikationen von thematischen Begrenzungsflächen eines Gebäudes.

Mögliche Anwendung

Die Bereitstellung von zusätzlicher Charakterisierung von Begrenzungsflächen eines Gebäudes erlaubt es, die Art der betrachteten Flächen näher zu spezifizieren. So können z.B. Flächen als Fassadenfläche oder Seitenwand klassifiziert werden. Auch ist es möglich, die Relevanz einer Gebäudefläche in den Daten zu hinterlegen. Diese Attribute können bei der automatischen Abstandsflächenberechnung für die Entscheidung über Erfordernis von Abstandsflächen zu Rate gezogen werden.
Zusätzliche Attribute über das Maß der baulichen Nutzung werden anhand der Gebäudegeometrie und thematischen Attribute, wie Anzahl der Vollgeschosse, etc. berechnet und bereitgestellt. Diese können dann mit den Regulierungen und Grenzwerten aus den Bebauungsplan-Objekten abgeglichen und auf Überschreitungen geprüft werden.
Es besteht die Möglichkeit dem Datenmodell die Berechnungsregeln als OCL hinzuzufügen. Dadurch kann bei der Validierung eines Instanz-Dokuments geprüft werden, ob die angegebenen Werte mit den vorliegenden Gebäudedatensatz konsistent sind.
Mittels des Werkzeugs *ShapeChange* können OCL-Ausdrücke nach Schematron übersetzt werden. Das resultierende Schematron-Dokument wird mittels eines Schematron-XSL (Extensible Stylesheet Language)-Stylesheets durch Anwendung eines XSLT (Extensible Stylesheet Language Transformation)-Prozessors in ein XSL-Dokument transformiert. Das Ergebnis-Dokument dient wiederum der Validierung des XML-Instanz-Dokuments (in diesem Fall das Dokument, welches das Gebäude enthält) mittels eines XSL-Prozessors (Robertson, 2003).

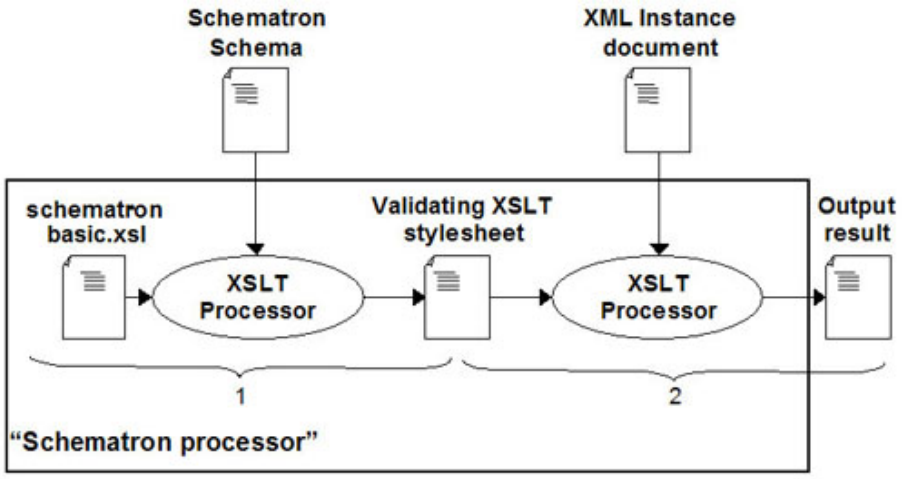


Abbildung 22 Schritte bei der Validierung eines XML-Instanz-Dokuments mittels Schematron (Robertson, 2003).

Durch die Erweiterung des ISO (International Organization for Standardization) Schematron Standards durch *Schematron QuickFixes* ist es zudem möglich, inkorrekte Werte im Instanz-Dokument direkt über im Schematron enthaltene sogenannte *Activity Elements* zu beheben (Kutscherauer, 2018). Diese Elemente enthalten Berechnungsregeln, die angewandt werden, wenn die Regelprüfung/Validierung des betrachteten Elements negativ ausfällt (Kutscherauer, 2018).

3.2.4 Package: RiskAssessment

Inhaltliche Beschreibung

Von der INSPIRE-Spezifikation wird die Thematik *Natural risk zones* übernommen (siehe (*D2.8.III.12 Data Specification on Natural Risk Zones – Technical Guidelines*, 2013)). Diese Thematik beschreibt Risikogebiete als Gebiete mit Population, kulturellen oder ökonomischen Wert, die natürlichen Gefahren wie Überflutungen, Erdbeben, Erdrutschen, Waldbrände, o.ä. ausgesetzt sind (*D2.8.III.12 Data Specification on Natural Risk Zones – Technical Guidelines*, 2013).

Die Anbindung an das CityGML-Modell erfolgt zum einen über die abstrakte Klasse *AbstractRiskZone*, welche als Subklasse der CityGML-Klasse *AbstractLogicalSpace*, modelliert wird. Zum anderen über die Gefahrenzone *AbtractHazardArea*, die sich von der Klasse *AbstractOccupiedSpace* ableitet.
Die Entscheidung für die jeweilige Superklasse zur Anbindung an das CityGML-Modell wird damit begründet, dass eine Risiko-Zone eine abstrakte Klassifizierung einer räumlichen Einheit darstellt. Im Gegensatz dazu hat ein gefährdendes Areal oder Phänomen direkte und physische Auswirkungen auf die Realität.

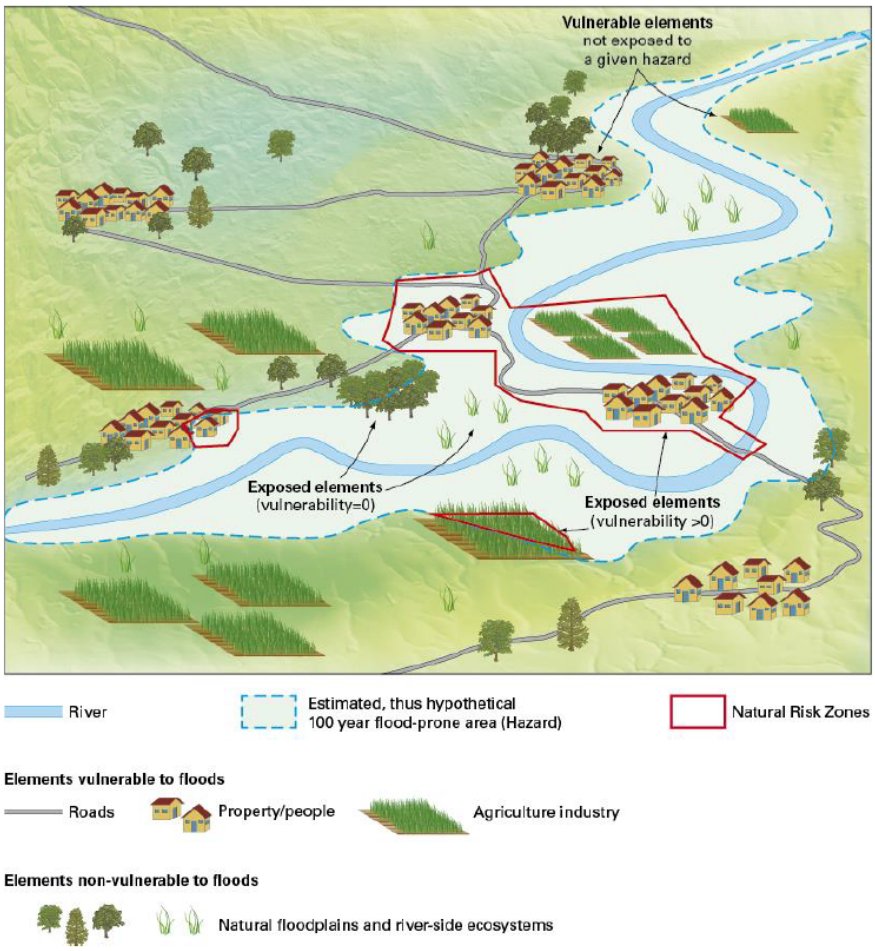


Abbildung 23 Visualisierung der INSPIRE-Thematik *Natural risk zones* (*D2.8.III.12 Data Specification on Natural Risk Zones – Technical Guidelines*, 2013).

Mögliche Anwendung

Beim Abschluss von Versicherungen im Kontext von Gebäuden können dem Lageplan Daten aus der Risikobewertung beigefügt werden, welche möglicherweise dazu dienen, den Versicherungsbeitrag des Objektes zu ermitteln. Um eine Schaden-Analyse im Falle eines Versicherungsfalls zu simulieren, können zusätzlich physikalische Eigenschaften (z.B. Material) der Objekte hinterlegt werden.

3.2.5 Package: RegulationConflict

Inhaltiche Beschreibung

Für die Modellierung von Konflikten zwischen Objekten und Regulierungen (z.B. aus dem Bebauungsplan) werden die Konzepte aus (Emamgholian et al., 2020) aufgegriffen. In dieser Veröffentlichung werden auftretende räumliche, semantische und zeitliche Konflikte in weiche und harte Konflikte kategorisiert. Weiche Konflikte sind durch Einfachheit und wenig Aufwand in der Datenanalyse charakterisiert, wohingegen harte Konflikte, ihrer Komplexität geschuldet, einen höheren Aufwand bei der Analyse erfordern (Emamgholian et al., 2020).
Es werden fünf Variablen definiert, welche bei der Einteilung von Konflikten in die Kategorien ”hart” und ”weich” angewandt werden:

- Anzahl der im Konflikt involvierten Regulierungen
  - Weich $=$ 1
  - Hart $\geq$ 2
- Anzahl der im Konflikt involvierten physikalischen Objekte
  - Weich $<$ 2
  - Hart $\geq$ 2
- LOD des geplanten Objekts
  - Weich $=$ LOD1
  - Hart $=$ LOD2 und LOD3
- LOD der Objekte in der Umgebung
  - Weich $=$ LOD1
  - Hart $=$ LOD2 und LOD3
- Räumliche Konfiguration der Regulierungen
  - Weich $=$ normale Solid-Geometrie
  - Hart $=$ irreguläre 3D-Formen (Linien, Flächen, etc.)

(Emamgholian et al., 2020)

Für den inspizierten Konflikt werden nach Bestimmung der fünf Variablen die Anzahl der als ”weich” oder ”hart” klassifizierten Variablen ermittelt. Die größere sich ergebenden Menge aus ”weichen” oder ”harten” Variablen definiert schließlich den Gesamtcharakter des Konflikts (Emamgholian et al., 2020).

Mögliche Anwendung

Um im Nachhinein einer Prüfung des Lageplans die Konflikte zwischen topographischen und konzeptionellen Objekten darstellen zu können, ist es möglich die Konflikt-Klassen der SiteplanADE zu verwenden. Diese sollen zum einen zur Visualisierung des Konflikts dienen, zum anderen Informationen über den Konflikt, wie z.B. involvierte topographische Objekte, involvierte Regulierungen, usw. (und so weiter), beinhalten (Emamgholian et al., 2020).
Die geometrische Ausprägung eines Konflikts lässt sich mit den standardmäßigen *AbstractSpace*-Geometrien des CityGML-*Core*-Moduls ausdrücken(wie z.B. in Abbildung 24).

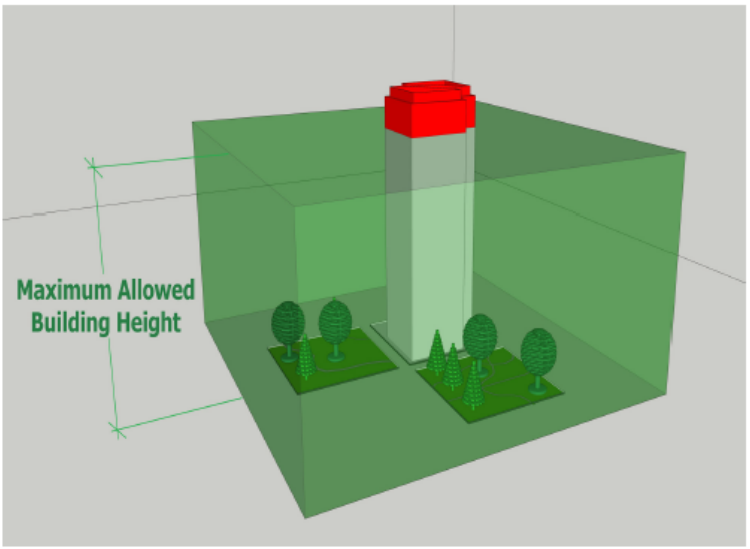


Abbildung 24 Beispielhafte Visualisierung eines Konflikts, bei welcher die maximal zulässige Gebäudehöhe überschritten wird (Emamgholian et al., 2020).

3.2.6 Package: LandUsePlanning

Package-Diagramm

Die Modellierung des Pakets *LandUsePlanning* hält sich an das in Unterabschnitt 3.1.3 in Punkt 2. vorgestellte Layer-Konzept der Modellierung von internationaler zu nationaler Ebene. In Abbildung 25 ist dargestellt, wie die verschiedenen Subpackages von dem CityGML-*Core*-Modell abgeleitet werden.

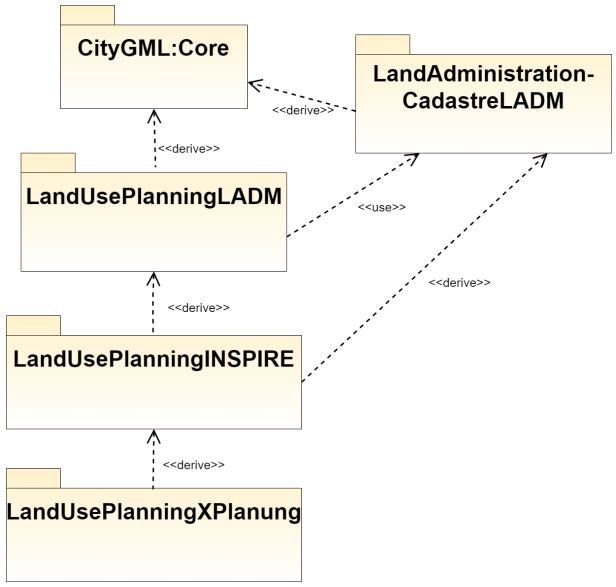


Abbildung 25 Package-Diagramm des Pakets LandUsePlanning.

Subpackage: LandUsePlanningLADM

Inhaltliche Beschreibung

Bei der Entwicklung der zweiten Edition des LADM wird das Modell u.a. auf die räumliche Planung ausgeweitet werden. Dabei wird auf die Standardisierung und Harmonisierung von digitalen und räumlichen Planwerken, wie z.B. in der INSPIRE-Initiative, reagiert (Lemmen et al., 2019).

Das Erweiterungspaket *Spatial Planning Information Package* des LADM wird für die Modellierung der internationalen Ebene von räumlicher Planung verwendet werden. Das Paket *LandUsePlanningLADM* wird über die Klasse *SP_PlanningUnit* als Subklasse von der *Core*-Klasse *Core::AbtstractLogicalSpace* mit dem CityGML-Modell verknüpft. Da sich die LADM-Modellierung von räumlicher Planung stark an den Klassen und am Konzept des LADM-Modell für Landadministration orientiert wird außerdem das LADM-Paket (siehe Abschnitt 3.2.7) genutzt (Lemmen et al., 2019).

Mögliche Anwendung

Die Klassen des Pakets *LandUsePlanningLADM* eignen sich für die internationale Abbildung von Objekten räumlicher Planung für Regionen in denen kein einheitlicher Standard für diese Thematik existiert. Zudem gewährleistet es die Darstellung von Daten aus verschiedenen Quellen im selben Modell. So können bspw. ALKIS-Daten aus Deutschland mit europäischen oder internationalen Daten dargestellt werden, wobei die ALKIS-Daten in ihrer ursprünglichen Form verwendet werden können. Es ist jedoch auch möglich, die unterschiedliche Daten auf derselben Modellierungsebene zu integrieren, was eine Generalisierung auf die höchste erforderliche, aber niedrigste notwendige Ebene, im Sinne der internationalen Reichweite, erforderlich macht. Dies kann z.B. in Grenzregionen von Vorteil sein, wenn länderübergreifende Daten fusioniert werden müssen.

Subpackage: LandUsePlanningINSPIRE

Inhaltliche Beschreibung

Wie in Unterabschnitt 3.1.3 in Punkt 2. beschrieben, ist die Thematik der räumlichen Planung in INSPIRE bereits modelliert. Diese INSPIRE-Thematik befindet sich im mittleren Modellierungslayer (siehe Abbildung 11). Für die Integration in die SiteplanADE werden kleinere Änderungen vorgenommen. Für die Spezifikation der einzelnen von INSPIRE übernommenen Datentypen sei auf die offizielle Dokumentation (*D2.8.III.4 Data Specification on Land Use - Technical Guidelines* (2013)) verwiesen. Der Grund der Übernahme vieler Klassen und Attribute liegt in der Schaffung einer einfachen Übertragung von einem Datenmodell in das andere, sowie Kompatibilität zwischen der SiteplanADE und der INSPIRE-Thematik.
Die Verknüpfung mit dem Paket *LandUsePlanningLADM* erfolgt zum einen über die Subklassen *ZoningElement* und *ZoningElement*, welche sich von der Klasse *LandUsePlanningLADM::SP_PlanningUnit* ableiten, zum anderen über die Klasse *OfficialDocumentation* als Subklasse von *LandAdminitstrationCadastre_LADM::LA_AdministrativeSource*. Dadurch, dass alle Klassen das räumliche Konzept von der CityGML-Klasse *Core::AbstractSpace* erben, wird das Modell um die volumetrische Darstellung erweitert.

Mögliche Anwendung

Durch die Nachahmung der INSPIRE-Thematik *Planned Land Use* lassen sich ohne großen Aufwand Daten dieses Standards in den Lageplan integrieren. Dies ermöglicht die Anwendung der SiteplanADE auch im europäischen Kontext.

Subpackage: LandUsePlanningXPlanung

Inhaltliche Beschreibung

Der XPlanung-Standard ist ein deutscher Standard für die Abbildung der räumlichen Planungswerke und befindet sich daher auf der untersten, nationalen Modellierungsebene (siehe Unterabschnitt 3.1.3 in Punkt 2.). Der Standard ist aufgrund der Begrenzung auf nationale Reichweite sehr detailliert und umfasst eine große Menge unterschiedlicher Klassen. Aus der Objektliste des BDVIs aus Unterabschnitt 3.1.1 geht hervor, dass für den amtlichen Lageplan v.a. Klassen aus dem Bebauungsplan-Paket des XPlanung-Standards von Relevanz sind.
Die Ankopplung an die Klassen *LandUsePlanningINSPIRE::ZoningElement* und *LandUsePlanningINSPIRE::SupplementaryRegulation* hält sich an die in Duan und Benner (2019) vorgeschlagenen Transformationsabbildungen zwischen den Klassen des XPlanung-Standards und den entsprechenden Klassen der INSPIRE-Thematik.
Eine Besonderheit ist, dass im XPlanung-Standard, trotz des Ziels der Enkodierung in XML, teilweise multiple Vererbung verwendet wird (XML erlaubt nur einfache Vererbung). Auch dies muss bei der Integration in die SiteplanADE beachtet werden. Neben der Ableitung von den INSPIRE-Klassen, erben auf der selben Ebene manche XPlanung-Klassen zusätzlich von anderen XPlanung-Klassen (siehe Abbildung 26). Um dies bei der Ableitung zu berücksichtigen, wird in der *ShapeChange*-Konfigurationsdatei die Regel *rule-xsd-cls-mixin-classes* hinzugefügt. Außerdem werden die XPlanung-Superklassen, welche bei der Mehrfach-Vererbung beteiligt sind, mit einem *taggedValue* mit dem Wert *gmlMixin = True* gekennzeichnet.
Dadurch, dass das räumliche Konzept von der CityGML-Klasse *Core::AbstractSpace* geerbt wird, wird das übernommene XPlanung-Modell um die volumetrische Darstellung erweitert.

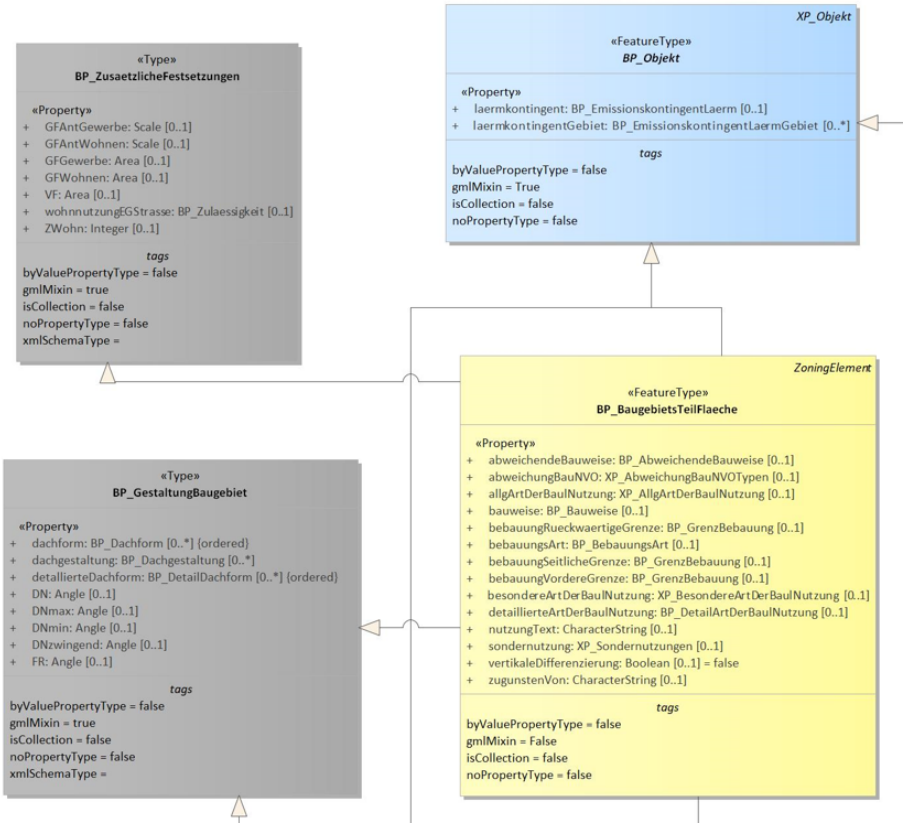


Abbildung 26 Auszug aus einem Klassendiagramm des Subpackages *LandUsePlanningXPlanung*. Die Klasse *BP_BaugebietsTeilflaeche* erbt sowohl von den Klassen *BP_ZusaetzlicheFestsetzungen, BP_GestaltungBaugebiet, BP_Objekt*, sowie der INSPIRE-Verknüpfungsklasse *ZoningElement*.

Mögliche Anwendung

Für die Modellierung von Lageplänen auf nationalem Level können Bebauungsplan-Datensätze im XPlanungGML-Format von den Gemeinden und Behörden bezogen werden und ohne großen Aufwand in fast unveränderter Form in einen Lageplan gemäß der SiteplanADE integriert werden.

Dadurch, dass alle Klassen das räumliche Konzept von der CityGML-Klasse *Core::AbstractSpace* erben, wird das XPlanung-Modell um die volumetrische Darstellung erweitert. Baugebietsflächen, mit Regulierungen bzgl. der maximalen Gebäudehöhe könnten damit als Solid-Geometrien mit entsprechender Maximalhöhe dargestellt werden (siehe Abbildung 27). Dies lässt zugleich Überschreitungen von Grenzwerten bestehender oder geplanter Gebäude überprüfen.

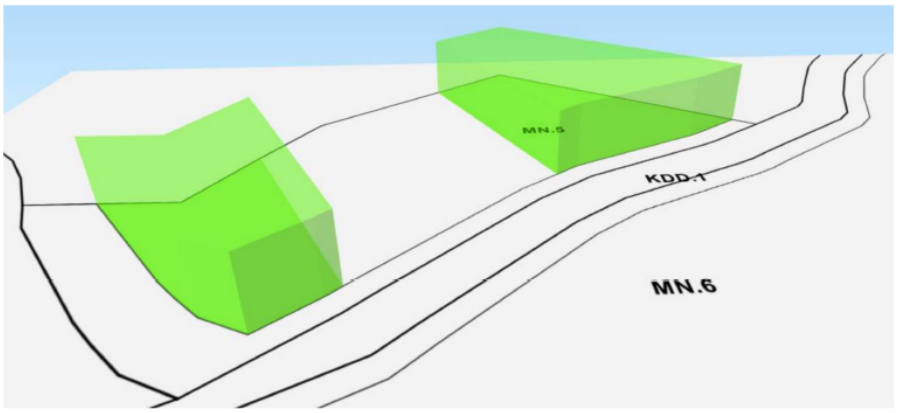


Abbildung 27 3D-Objekte eines Raumplanungswerk, welches Baugebiete mit maximaler Gebäudehöhe mittels Solid-Geometrien implementiert (Bydłosz et al., 2018).

3.2.7 Package: LandAdministrationCadastre

Package-Diagramm

In Abbildung 28 sind die Abhängigkeiten der verschiedenen Pakete auf den verschiedenen Modellierungsebenen dargestellt (vgl. (vergleiche) Unterabschnitt 3.1.3 in Punkt 2.).

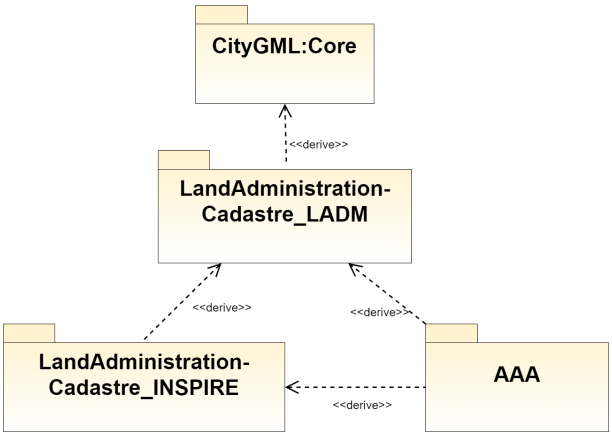


Abbildung 28 Package-Diagramm des Pakets LandAdministrationCadastre.

Subpackage: LandAdministrationCadastre_LADM

Inhaltliche Beschreibung

Das Modell des LADM-Standards stellt Klassen zur Beschreibung von Beziehungen zwischen Parteien und Land zur Verfügung (Lemmen et al., 2015).
Die Anbindung des LADM-Standards erfolgt über die Klasse *LA_BAUnit*. Diese Klasse stellt eine Buchungseinheit im Eigentumsregister dar und gruppiert bzw. fasst Objekte geographischer Ausdehnung in dieser Buchungseinheit zusammen (Lemmen, 2012). Die *LA_BAUnit*-Klasse lässt sich daher von der *CityObjectGroup::CityObjectGroup* ableiten, da diese eine logische Gruppierung von Objekten erlaubt. Die Klasse *LA_SpatialUnit* für die Repräsentation von mit Rechten belasteten räumlichen Objekten wird bereits über das räumliche Konzept vom CityGML-Modell implizit bereitgestellt.

Objekte können dann über die *LA_BAUnit*-Klasse als Subklasse von *CityObjectGroup::CityObjectGroup* zu einer administrativen Einheit gruppiert werden. Damit wird die Kompatibilität der Modellierung zwischen der SiteplanADE und dem LADM gewährleistet.

Mögliche Anwendung

Über die Adaption an den LADM-Standard wird es möglich, auf internationaler Ebene Eigentumsverhältnisse abzubilden. Dies kann sinnvoll sein, wenn auf keine vorhandenen Standards zurückgegriffen werden kann, oder der LADM-Standard der kleinste gemeinsame Nenner für die Zusammenführung von Datensätzen unterschiedlicher Datenmodelle bildet.

Subpackage: LandAdministrationCadastre_INSPIRE

Inhaltliche Beschreibung

Die INSPIRE-Thematik *CadastralParcel* basiert auf dem LADM (Lemmen et al., 2019). Die beiden Datenmodelle sind daher zueinander kompatibel (*D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines*, 2014). Jedoch bildet die INSPIRE-Thematik nur den geographischen Teil des Katasters ab (*D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines*, 2014; Lemmen et al., 2019). Die Modellierung von Rechten und Eigentum verschiedener Parteien in Bezug auf Grund- oder Flurstücke wird allerdings von der INSPIRE-Thematik *CadastralParcel* nicht abgedeckt (*D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines*, 2014). LADM bildet daher einen breiteren Kontext für die INSPIRE-Thematik (*D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines*, 2014). Die INSPIRE-Thematik kann demnach als Profil des LADM aufgefasst werden, welche das LADM auf die räumliche Einheit des Flurstücks (*CadastralParcel*) und eine bestimmte Geometrie beschränkt (Laurent, 2017).
Die INSPIRE-Klasse *CadastralParcel* stellt das Äquivalent zur LADM-Klasse *LA_SpatialUnit* dar (Seifert, 2012). Deshalb wird diese nur im LandAdministrationCadastre_INSPIRE-Paket als INSPIRE-Klasse modelliert und als Subklasse der CityGML-Klasse *Core::AbstractLogicalSpace* an das CityGML-Modell angehängt. Die INSPIRE-Klasse *CadastralZoning* bildet eine logische Gruppierung von Flurstücken leitet sich somit von der LADM-Klase *LA_SpatialUnitGroup* ab.

Klassen für die Abbildung von Information über die Bodenbeschaffenheit sind in der INSPIRE-Thematik *Soil* enthalten. Die INSPIRE-Klasse *DerivedSoilProfile* wird über eine Assoziation der Subklasse *SoilProperties*, welche sich von *ADEOfLandUse* ableitet, mit CityGML verknüpft. Ein Bodenprofil (*SoilProfile*) stellt einen Querschnitt des Untergrundes bis zu einer bestimmten Tiefe dar und gibt Aufschluss über die Beschaffenheit des untersuchten Bodens (*D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines*, 2013).

Die CityGML-Klasse *LandUse* ist durch das Halten von der Bodenprofil-Information das Äquivalent zur INSPIRE-Klasse *SoilBody*. Ein Bodenkörper (*SoilBody*) ist ein geographischer Bereich bestehend aus einer Anzahl in diesem Bereich gefundener Bodenprofile.

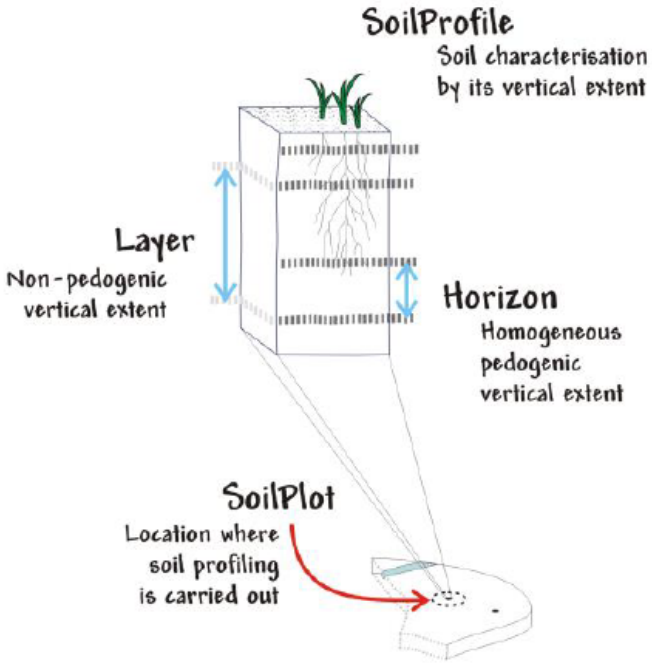


Abbildung 29 Schematische Visualisierung des Bodenprofil-Konzeptes aus der INSPIRE-Spezifikation (*D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines*, 2013).

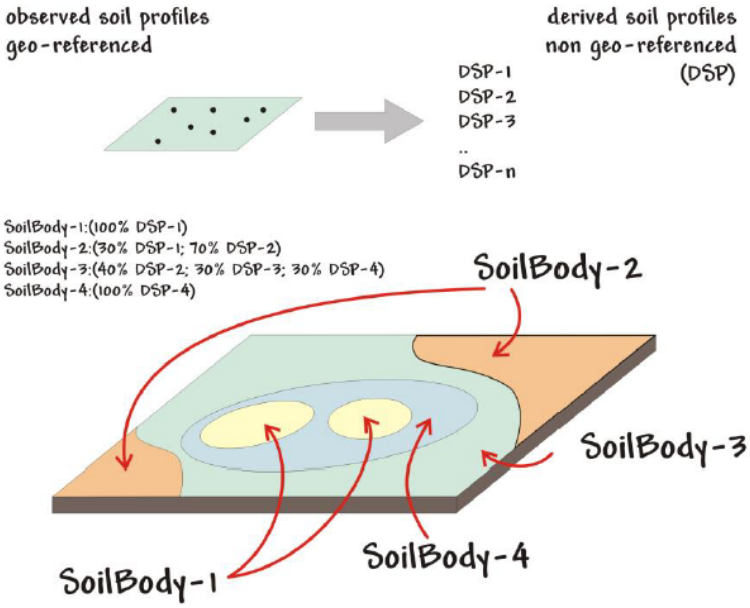


Abbildung 30 Der Bodenkörper besteht aus einer Anzahl abgeleiteter Bodenprofile (*D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines*, 2013).

Mögliche Anwendung

Durch die Übernahme der Klassen der INSPIRE-Thematik *CadastralParcel* können Daten aus der Open-Data-Initiative INSPIRE für die Erstellung eines SiteplanADE-Lageplans ohne Datentransformation, sondern über einfaches Mapping auf die Output-XML-Objekt-Templates in einer FME, integriert werden.

Subpackage: LandAdministrationCadastre_AAA

Inhaltliche Beschreibung

Das Subpackage *LandAdministrationCadastre_AAA* beinhaltet diejenigen Klassen aus dem offiziellen AAA-Modell welche in der Objektliste des BDVI (siehe Unterabschnitt 3.1.1) geführt werden.
Die Klassen des Pakets *LandAdministrationCadastre_AAA* leiten sich von kompatiblen Klassen der INSPIRE-Thematik ab (z.B. *AX_Flurstueck* → *CadastralParcel* oder *AX_Gemarkung* → *CadastralZoning*). Wenn keine korrespondierenden Klassen vorhanden sind (wie bspw. für Vermessungspunkt-Klassen *AX_Netzpunkt, AX_Grenzpunkt*, etc.) werden AAA-Klassen von den korrespondierenden LADM-Klassen (im Falle von Vermessungspunkt-Klassen von der Klasse *LA_Point*) abgeleitet.
Die AAA-Klasse *AX_Bodenschaetzung* leitet sich von der INSPIRE-Klasse *DerivedSoilProfile* ab.

Mögliche Anwendung

Durch die Kompatibilität der SiteplanADE am AAA-Standard können Daten von den Verwaltungsinstitutionen bezogen werden und direkt auf die entsprechenden SiteplanADE-Klassen abgebildet werden, ohne Transformationen durchführen zu müssen.

3.2.8 Package: ConstructionLaw

Inhaltliche Beschreibung

Das Paket *ConstructionLaw* liefert Klassen für die Abbildung von bau-rechtlichen Objekten mit Raumbezug. Da es sich dabei um konzeptionelle Objekte ohne physikalische Eigenschaften handelt, leiten sich die Klassen von der CityGML-*Core*-Klasse *AbstractLogicalSpace* ab. Mittels des ADE-*Hooks* können über die *ADEOfAbstractCityObject*-Unterklasse *ConstructionLawProperties* Instanzen der abstrakten Superklasse *AbstractContructionLawObject* mit jeglichen CityGML-Objekten assoziiert werden.
Zusätzlich werden Klassen der verschiedenen Parteien, welche mit einem (offiziellen) Lageplan rechtlich in Verbindung gebracht werden können, bereit gestellt. Dies sind u.a. der*die Lageplanverfasser*in, die Gemeinde, die Nachbarn, der*die Antragssteller*in, sowie die Bauaufsichtsbehörde (Wirth & Schneeweiß, 2019).

Mögliche Anwendung

Im konkreten Fall des amtlichen Lageplans können die Abstandsflächen bzw. -räume nach BauO über die Klasse *ClearanceSpace* (Subklasse der abstrakten Klasse *AbstractConstructionLawObject*) modelliert werden. Dabei können für die Abstandsflächen der einzelnen Gebäudekomponenten flächenhafte Geometrien verwendet werden. Für den resultierenden Abstandsraum eines Gebäudes würden alle Gebäudekomponenten mit den Abstandsflächen auf die x-y-Ebene projiziert, vereinigt und anschließend zu einer Solid-Geometrie extrudiert werden.

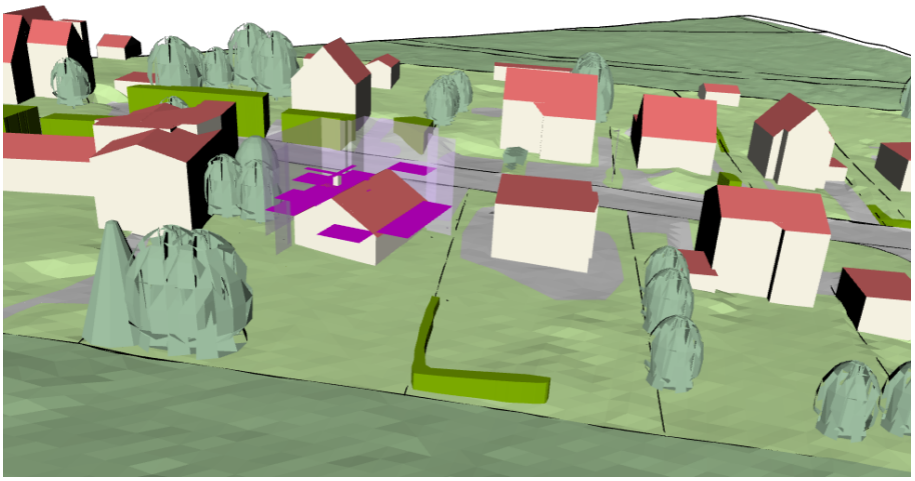


Abbildung 31 Darstellung der Abstandsflächen, welche sich aus den einzelnen Gebäudekomponenten ergeben, sowie des resultierenden Abstandsraumes.

3.2.9 Package: ThematicGrouping

Inhaltliche Beschreibung

Das Paket *ThematicGrouping* stellt Klassen zur logischen Aggregation von CityGML-Objekten zur Verfügung und leitet sich daher von der CityGML-Klasse *CityObjectGroup* ab. Die thematische Aufteilung der Objekte hat das Layer-Prinzip im GIS-Bereich als Vorlage. Die erste generelle thematische Aufteilungen kategorisiert Objekte in die Gruppen der existierenden Objekte (*ExisitingObjects*) und geplanten Objekte (*PlannedObjects*). Weiter werden die geplanten und existierenden Objekt-Gruppen jeweils in topographische und konzeptionelle Objekt-Gruppen unterteilt.

Mögliche Anwendung

Die Nachahmung des Layer-Prinzips aus dem Bereich der GIS-Software erlaubt es bei geeigneter Software, durch Ein- und Ausblenden von bestimmten Objekt-Gruppierungen sich einen Überblick und Vergleich zwischen geplanten Objekten und dem Ist-Zustand zu verschaffen. Auch können vereinfacht Berechnungen auf heterogene Gruppen von Objekten angewandt werden. Ohne die Vorgruppierung im Datensatz wird dies sonst bereits bei der Auswahl der Objekte (bspw. über XML-Abfragen wie XPath oder XQuery) erschwert, da die Abfragen mit steigendem Grad der Heterogenität der Objekt-Auswahl immer komplexer werden.

4 Automatisierte Abstandsflächenberechnung

4.1 Vorarbeit

4.1.1 Abstandsflächen im öffentlichen Baurecht

Sinn und Nutzen der Abstandsflächen

Abstandsflächen sind Flächen vor den Außenwänden von baulichen Anlagen, die gesetzlich von oberirdischen Gebäuden oder anderen Anlagen von denen eine Wirkung wie von Gebäuden ausgehen freigehalten werden müssen (BauO NRW §6 Absatz 1 mit Stand vom 24.7.2020). Vorrangig dienen Abstandsflächen der Wahrung des sozialen Friedens zwischen Nachbarn (U. Krause, 2013). Sie sollen für ausreichend Belichtung, Belüftung und Sonnenbestrahlung der Wohnräume sorgen (U. Krause, 2013).

Bedingung für Abstandsflächen bei baulichen Anlagen

Gebäude und andere Anlagen gegenüber von Gebäuden und Grundstücksgrenzen erfordern Abstandsflächen, wenn diese eine Höhe von mehr als 2 m über der Geländeoberfläche aufweisen und von ihnen eine Gebäude-ähnliche Wirkung ausgeht (BauO NRW §6 Absatz 1 mit Stand vom 24.7.2020). Außerdem Anlagen, welche es erlauben von Menschen betreten zu werden und höher als 1 m über der Geländeoberfläche sind (BauO NRW §6 Absatz 1 mit Stand vom 24.7.2020). Ausnahmen, bei denen keine Abstandsflächen erforderlich sind, sind Situationen, wenn Außenwände an die Grundstücksgrenzen errichtet werden und nach planungsrechtlichen Vorschriften an die Grenze gebaut werden muss, oder es erlaubt ist, an die Grenze zu bauen, wenn es mit Sicherheit festgestellt ist, dass auf dem benachbarten Grundstück ohne Grenzabstand gebaut wird (BauO NRW §6 Absatz 1 mit Stand vom 24.7.2020).

Berechnung der Abstandsflächentiefe nach Landesbauordnung

Die Berechnung der Abstandsflächen ist in der Landesbauordnung geregelt, sie kann sich je nach Bundesland unterscheiden. Das Vorgehen bei der Berechnung ist in der entsprechenden BauO im Paragraphen der Abstandsflächen zu finden.
Im Falle der Abstandsflächenberechnung nach BauO NRW §6 wird bei der Berechnung wie folgt vorgegangen:

1. **Bestimmung der Außenwandhöhe:** Gemessen von der Geländeoberfläche bis zur Schnittlinie der Wand mit der Dachhaut, oder bis zum oberen Abschluss der Wand
   (Für Außenwände, bestehend aus Wandteilen, wird die Höhe für jedes Wandteil bestimmt; bei geneigter Geländeoberfläche wird ein Mittelwert der Wandhöhe an den vertikalen Begrenzungen der Wand bzw. des Wandteils ermittelt)

2. **Hinzurechnen zur Wandhöhe:**

   - Mit Faktor 1:
     - Die Höhe von Dächern und Dachteilen mit einer Dachneigung von mehr als 70 Grad
     - Die Höhe von Giebelflächen im Bereich von Dächern und Dachteilen, wenn die Dachneigung mehr als 70 Grad beträgt
   - Mit Faktor $\frac{1}{3}$:
     - Die Höhe von angrenzenden Dächern und Dachteilen mit einer Dachneigung von mehr als 45 Grad
     - Die Höhe von angrenzenden Giebelflächen im Bereich von Dächern und Dachteilen, wenn die Dachneigung mehr als 70 Grad beträgt
     - Die Höhe von angrenzenden Dächern mit Dachgauben oder Dachaufbauten, wenn die Gesamtlänge je Dachfläche einen Betrag von mehr als die Hälfte der sich darunter befindenden Gebäudewand aufweist

   Die resultierende Größe ist die Wandhöhe H.

3. **Berechnung der Abstandsflächentiefe T:**

   - $T = a \cdot H$, mit der Bedingung $T \geq 3\,m$
   - Werte von a:
     - $a = 0.4$
     - $a = 0.2$ in Gewerbe- und Industriegebieten, in Kerngebieten zu öffentlichen Verkehrs-, Grün- und Wasserflächen
   - $T = 3\,m$ in Wohngebieten für Gebäude der Gebäudeklasse 1 und 2 mit nicht mehr als drei oberirdischen Geschossen

Die resultierende Formel für die Berechnung der Abstandsflächentiefe T ergibt sich demnach zu:

$$T = a \cdot [0.5 \cdot (\underbrace{h(P_0) - DGM(P_0')}_{h_0} + \underbrace{h(P_1) - DGM(P_1')}_{h_1}) + \Delta H(\alpha, h_D, \psi)] \tag{4.1}$$

Dabei gibt die Funktion $h(P_i)$ die Höhe des oberen Abschluss der vertikalen Begrenzung des Wandteils zurück. Die Funktion $DGM(P_i')$ bestimmt die Geländehöhe an der Position von P_i' anhand des DGMs, welcher den auf die x-y-Ebene projizierten Punkt P_i darstellt. $\Delta H(\alpha, h_D)$ bestimmt die Zurechnung der Dachhöhe h_D in Abhängigkeit der Dachneigung α und Konfiguration der Dachaufbauten ψ. Die Dachhöhe bezieht sich dabei nicht auf die Geländeoberfläche, sondern relativ zum oberen Abschluss der Wand bzw. des Wandteils.

a ist der Faktor, der nach der Berechnung der Wandhöhe angebracht wird und sich nach Art des Baugebiets richtet, in der die Abstandsfläche liegt.
Die Länge L der Abstandsfläche ergibt sich über die horizontale Strecke zwischen den oberen (oder unteren) Abschlüssen der vertikalen Begrenzungen der Wand bzw. des Wandteils. In Abbildung 32 sind die in Gleichung 4.1 enthaltenen Parameter visualisiert.

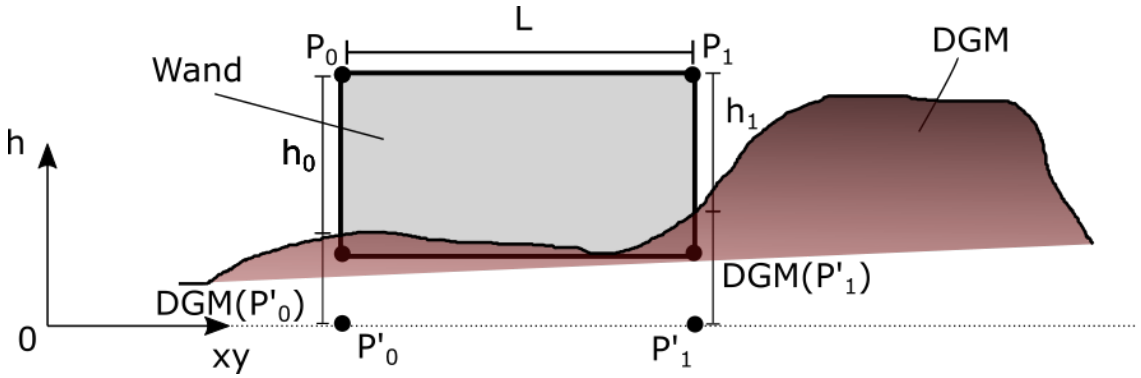


Abbildung 32 Darstellung der Parameter in Gleichung 4.1 für die Abstandsflächentiefenberechnung.

Gebäudekomponenten ohne Abstandsflächen

Nach BauO NRW §6 Absatz 6 mit Stand vom 24.7.2020 werden Bauteile (z.B. Gesimse, Dachüberstände), die nicht mehr als 1.5 m vor die Außenwand treten, von der Abstandsflächenpflicht befreit. Ebenfalls Vorbauten, wenn sie insgesamt nicht mehr als ein Drittel der Breite der jeweiligen Außenwand belegen, nicht mehr als 1.6 m aus dieser Wand heraustreten und mindestens 2 m von der gegenüberliegenden Nachbargrenze entfernt bleiben. Des weiteren werden Seitenwände von Vorbauten und Dachaufbauten von Gebäuden an der Grundstücksgrenze, auch wenn sie nicht an der Grundstücksgrenze errichtet werden, nicht bei der Abstandsflächenberechnung berücksichtigt. Auch Garagen und Gebäude ohne Aufenthaltsräume, sowie Tiefgaragenzufahrten, Aufzüge zu Tiefgaragen und Feuerstätten mit bis zu 30 m^3 Rauminhalt sind bei einer Wandhöhe nicht größer als 3 m zu vernachlässigen.
In BauO NRW §6 Absatz 7 mit Stand vom 24.7.2020 ist zudem festgelegt, dass Anlagen zum Zweck der Energieeinsparung, sowie Solaranlagen an bestehenden Gebäuden keine Abstandsflächen erforderlich machen, wenn sie

- sich in einem Abstand von mindestens 2.5 m von der Nachbargrenze befinden,
- eine Stärke von 0.25 m oder weniger aufweisen.

Weiter wird in BauO NRW §6 Absatz 9 mit Stand vom 24.7.2020 definiert, dass bei Änderungen von vor dem 1.Januar 2019 errichteten Gebäuden mit Wohnungen, Aufzüge, welche vor die Außenwand treten bei der Abstandsflächenberechnung vernachlässigt werden, wenn sie

- nicht länger als 2.5 m sind,
- nicht höher 0.5 m über dem oberen Abschluss des obersten angefahrenen Geschosses mit Wohnungen sind,
- höchstens 2.5 m aus der Außenwand hervortreten und
- nicht weniger als 1.5 m vor der Nachbargrenze entfernt sind.

4.1.2 Eingangsdaten für die automatisierte Berechnung von Abstandsflächen

Definition der Eingangsdaten

Neben der geltenden BauO, in der die mathematischen Regeln für die Berechnung der Abstandsflächentiefe definiert sind, werden in BauO NRW §6 Absatz 4 implizit weitere Eingangsdaten genannt. Um die Abstandsflächentiefe automatisiert berechnen zu können, müssen Wandkomponenten, sowie Dachflächen des Gebäudemodells vorliegen. Um diese aus einem digitalen Gebäudemodell zu extrahieren, ist das Modell in einem Datenmodell erforderlich, dass eine semantische Filterung nach Gebäudekomponenten erlaubt. So können Wandflächen, Dachflächen, Gebäudeinstallationen, etc. für weitere Analysen, welche die Abstandsflächenberechnung betreffen, aus dem Gebäudemodell über thematische Abfragen extrahiert werden. Um die Höhe der Außenwand über Grund zu bestimmen, sind (virtuelle) Schnitte der Wandflächenobjekte mit dem Gelände zu berechnen. Somit ist für die Berechnung der Außenwandhöhe ein DGM erforderlich. Um Gebäudekomponenten bzgl. ihres Abstands zur Nachbargrenze untersuchen zu können, ist zudem ein Datensatz der Grundstücksgrenzen notwendig. Auch spielen Baugebiet, öffentliche Flächen und die Gebäudeklasse und Anzahl der überirdischen Vollgeschosse des geplanten Objekts ein Rolle in der Berechnung der Abstandsflächen.
Zusammenfassend werden folgende Daten und Informationen für die automatisierte Abstandsflächenberechnung benötigt:

- 3D-Gebäudemodell
- Gebäudeklasse und überirdischen Vollgeschosse
- Ankerpunkt und Rotationswinkel
- DGM
- Mathematische Berechnungsregeln und Parameter nach BauO
- Grundstücksgrenzen (aus Kataster)
- Art der baulichen Nutzung des Baugebiets des Gebäudes (aus Bebauungsplan)
- Öffentliche Verkehrs-, Grün- und Wasserflächen (aus Bebauungsplan)

3D-Gebäudemodell

Datenherkunft

Das Gebäudemodell wird im Zuge der Zusammenstellung für die Generierung des (amtlichen) Lageplans von dem*der Entwurfsverfasser*in bezogen.

Relevante Gebäudekomponenten

Für die Abstandsflächen relevante Gebäudekomponenten wurden in Unterabschnitt 4.1.1 im Zuge der Vorstellung der Abstandsflächen im öffentlichen Baurecht genannt:

- Wände
- Dach
- Gebäudeinstallationen (Vorbauten, Dachaufbauten, Gesimse, etc.)

Datenmodell des 3D-Gebäudemodells

Für die schnelle und unkomplizierte Extraktion der Gebäudekomponenten aus dem Gebäudemodell, die für die Berechnung der Abstandsflächen relevant sind (Außenwände, Dachflächen und Gebäudeinstallationen), ist, neben der geometrischen, auch eine semantische Modellierung des Gebäudes erforderlich. Dies erlaubt eine semantische Filterung des Gebäudemodells bzgl. der Objektmenge, die für die Abstandsflächenberechnung notwendig ist. Ein Datenmodell, welches diese semantische Information neben der geometrischen Repräsentation in sich vereint, ist das CityGML-Datenmodell (siehe Abbildung 33).

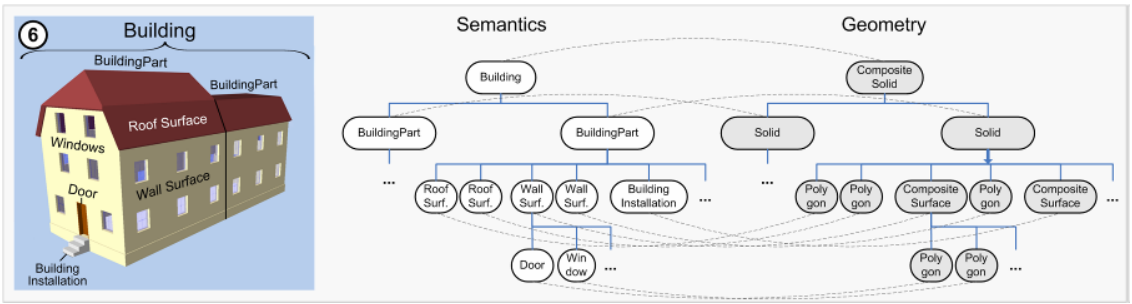


Abbildung 33 Darstellung der in CityGML zusammengefassten kohärenten Modellierung von Semantik und Geometrie (Stadler & Kolbe, 2007).

Ein weiterer Gesichtspunkt ist, dass der CityGML-Standard es erlaubt, das Gebäude mittels Volumen-begrenzenden thematischen Flächen zu modellieren. So ist die thematische CityGML-Klasse *WallSurface* eine die von außen sichtbare äußere flächenhafte Begrenzung eines Gebäudes, eines Gebäudeteils oder einer Gebäudeinstallation (Special Interest Group 3D, 2018e).
Genauso verhält es sich mit der Klasse *RoofSurface*. Dies modelliert die obere äußere Begrenzungsfläche eines Gebäudes, eines Gebäudeteils oder einer Gebäudeinstallation (Special Interest Group 3D, 2018f). Wird der Gesetzestext für Berechnung der Wandhöhe betrachtet, so sind die Formulierungen *"Außenwand"* und *"[...] Schnittlinie der Wand mit der Dachhaut [...]"* (BauO NRW §6 Absatz 4 mit Stand vom 24.7.2020) zu finden.

Der erste Auszug besagt, dass nur die Außenwände des Gebäudes für die Berechnung Relevanz haben. Die Definition der CityGML-Klasse *WallSurface* beinhaltet genau diese Bedingung der äußeren Begrenzungsfläche des Gebäudes.

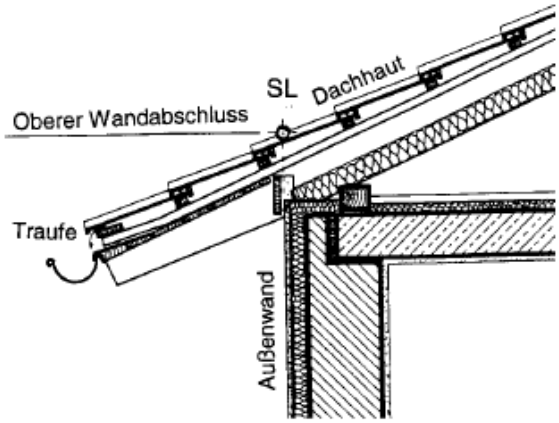


Abbildung 34 Darstellung der Schnittlinie *SL* des oberen Wandabschluss mit der Dachhaut (Noack et al., 2005).

Die zweite Formulierung erfordert es, die Modellierung eines Gebäudes in CityGML näher zu betrachten. Es gilt, dass die CityGML-Klassen *GroundSurfaces, WallSurfaces, RoofSurfaces, OuterFloorSurfaces, OuterCeilingSurfaces, ClosureSurfaces, Doors und Windows*, oder Referenzen zu diesen, das Volumen des Gebäudemodells begrenzen (siehe Abbildung 35) (Special Interest Group 3D, 2018d). D.h., diese Volumenbegrenzung impliziert, dass der Volumenkörper (*Solid*) geschlossen ist (”es kann kein Wasser von Innen nach außen fließen”) (Special Interest Group 3D, 2018a). Eine Begrenzungsfläche besteht aus einem Set von Polygonen (Special Interest Group 3D, 2018b). Ein Polygon ist planar und liegt in einer Ebene (Special Interest Group 3D, 2018c). Begrenzungsflächen weisen deshalb, aufgrund ihrer geometrischen Definition, kein Volumen auf. D.h., die Adjazenz einer Wandfläche mit einer Dachfläche ist die Berührung der äußeren Begrenzungsflächen, in anderen Worten, der äußeren Wand mit äußerer Dachhaut. Die Höhe dieser Berührung(-slinie) erfüllt folglich die im Gesetzestext geforderte Definition der Wandhöhe.

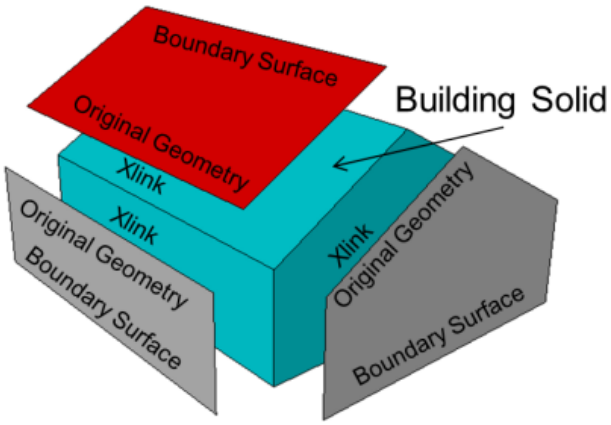


Abbildung 35 Die thematischen Flächen *RoofSurface* und *WallSurface* begrenzen den Volumenkörper (*Solid*) des Gebäudemodells (Special Interest Group 3D, 2018d).

Aufgrund der vorherigen Ausführungen ist es ersichtlich, dass die Modellierung des Gebäudes im CityGML-Format die automatisierte Berechnung der Abstandsflächen erleichtert. Die im Gesetz geforderten Außenflächen, sowie der Schnitt von Wandflächen mit der äußeren Dachhaut, sind infolge der Volumen-begrenzenden Modellierung im CityGML-Standard implizit vorhanden. Ein valides CityGML-Gebäudemodell erfordert deshalb keine Bearbeitung bzgl. seiner Geometrie, sondern kann direkt für die Abstandsflächenberechnung verwendet werden.

Ein weiterer Grund für die Abstandsflächenberechnung am CityGML-Gebäudemodell ist, dass im Zuge der Erstellung des 3D-Lageplans im CityGML-Format ohnehin eine Konvertierung des architektonischen Modells nach CityGML durchgeführt wird. Auch die Abstandsflächen des Gebäudes sollen im Lageplan dargestellt werden. Um Konsistenz zwischen dem Gebäudemodell im Lageplan und den berechneten Abstandsflächen zu gewährleisten, sollte die Berechnung am selben Modell durchgeführt werden, da nicht ausgeschlossen werden kann, dass wegen Schwierigkeiten bei der Transformation zwischen Architekturmodellen (z.B. IFC) und CityGML geometrische und semantische Abbildungsungenauigkeiten und Informationsverlust auftreten (Gilbert et al., 2020; de Laat & van Berlo, 2010; Kolbe, 2012; Stouffs, Tauscher & Biljecki, 2018; Kaden & Kolbe, 2016; Herle et al., 2020; Mekawy, 2010; El-Mekawy, Östman & Shahzad, 2010; Donkers, 2013; Donkers et al., 2015; Floros et al., 2018; Nagel, Stadler & Kolbe, 2009; Ohori et al., 2018; Salheb, 2019).

Neben der semantischen Abbildung erweist sich v.a. auch die geometrische Transformation zwischen den Datenmodellen BIM und CityGML als problematisch, da sich der geometrische Modellierungsansatz der beiden Datenmodelle unterscheidet (Herle et al., 2020). So werden BIM-Geometrien (neben der B-Rep (Boundary Representation)) als CSG (Constructed Solid Geometry) bzw. *Swept Solid* definiert, wobei volumetrische Körper mittels boolescher Operatoren miteinander verschnitten bzw. Flächen-Geometrien entlang Extrusionspfaden definiert werden (Herle et al., 2020; Gilbert et al., 2020; Donkers et al., 2015).

Diese Geometrie-Repräsentation im BIM-Datenmodell ist implizit, es werden nur die Parameter gespeichert, die für die Rekonstruktion der Geometrie-Objekte notwendig sind (Donkers et al., 2015). In CityGML wird die B-Rep für die Modellierung von Geometrie unterstützt (Herle et al., 2020; Gilbert et al., 2020). D.h., Geometrien werden anhand von Punkten, Kanten und Oberflächen, welche mit Koordinaten spezifiziert werden, abgebildet (Herle et al., 2020).

Diese, aus den unterschiedlichen Modellierungsansätzen resultierenden, Transformationsungenauigkeiten zwischen den Datenmodellen können die Abstandsflächenberechnung beeinflussen und bewirken, dass die Abstandsflächen, welche anhand des Gebäudemodells in einem anderen Datenmodell als CityGML erhoben wurden, anschließend nicht in erforderlicher Weise mit dem nach CityGML transformierten Gebäudemodell zusammenpassen.

Level Of Detail des Gebäudemodells

Es muss des weiteren in Betracht gezogen werden, welches LOD das CityGML-Gebäudemodell für die Berechnung der Abstandsflächen aufweisen muss. Das LOD-Konzept im CityGML Standard gibt die Auflösung bzw. den Detailgrad der geometrische Darstellung eines Objektes an (Gröger et al., 2012; Kolbe et al., 2020). In Tabelle 7 werden die verschiedenen Anforderungen an die verschiedenen LODs des CityGML 2.0-Standards tabellarisch dargestellt. Wie schon in *Datenmodell des 3D-Gebäudemodells* beschrieben, ist nur die äußere Hülle des Gebäudes für die Berechnung der Abstandsflächen von Relevanz.
LOD3 wird als Detaillierungsgrad entsprechend architektonischer Modelle angesehen und impliziert die Darstellung aller möglichen Details der äußeren geometrischen Form eines Objektes in hoher Genauigkeit (Gröger et al., 2012; Kolbe et al., 2020). Zu beachten ist, dass in der CityGML Version 3.0 LOD4 nicht mehr vorhanden ist. Wie in Tabelle 7 beschrieben, wird LOD4 für die Repräsentation des Innenraums von Objekten (z.B. von Gebäuden) verwendet (Kutzner et al., 2020). Diese Innenraumobjekte können in CityGML 3.0 in den verbliebenen LOD0-3 dargestellt werden (Kutzner et al., 2020). Somit ist LOD3 die höchste Detaillierungsstufe für die geometrische Darstellung von Objekten.

Um den benötigten LOD des Gebäudemodells bestimmen zu können, muss auf die relevanten Gebäudekomponenten in Abschnitt 4.1.2 Rücksicht genommen werden. Es ist ersichtlich, dass die Genauigkeitsanforderungen an die Abstandsflächen hoch sind, da eine fehlerhafte oder ungenaue Lage der Abstandsflächen direkte Auswirkungen auf die Realisierbarkeit und Genehmigung des Bauvorhabens haben können. Daraus resultiert, dass auch für die geometrische Form des Gebäudemodells diese Genauigkeitsanforderung gilt, da sich die Genauigkeit der Komponenten wie Wände, Dach und Gebäudeinstallationen direkt auf die resultierende Abstandsfläche auswirken bzw. deren Maße (z.B. das Verhältnis der Breite eines Vorbaus in Bezug auf die Außenwand) auch darüber entscheiden können, ob sie relevant für die Abstandsflächenberechnung sind, oder nicht.
D.h., es muss ein LOD gewählt werden, dass es erlaubt die Geometrie in entsprechend hoher Genauigkeit darzustellen. Hohe Genauigkeit ist nach Tabelle 7 ab dem LOD3 Voraussetzung. Außerdem muss es möglich sein, neben Wänden und Dach auch Gebäudeinstallationen zu modellieren. Auch dies ist ab dem LOD3 möglich. Objekte des Gebäudeinnenraums, wie bei LOD4 zu finden, spielen hingegen bei der Berechnung der Abstandsflächen keine Rolle.
Auf Tabelle 7 verweisend, wird daher der notwendige LOD eines Gebäudemodells für die Abstandsflächenberechnung auf LOD3 festgelegt.
Da in CityGML 3.0 für Gebäudemodelle auf Bauwerksebene LOD3 empfohlen wird, wird ebenfalls für Datensätze im CityGML 3.0-Format für die Abstandsflächenberechnung LOD3 gefordert (Kolbe et al., 2020). Da für beide Versionen von CityGML dasselbe LOD festgelegt wird, bedeutet das zudem Kompatibilität zwischen dem neuen LOD-Konzept in CityGML 3.0 und dem alten LOD-Konzept in CityGML 2.0. Für neue Datensätze im CityGML 3.0-Format, sowie für alte Datensätze im CityGML 2.0-Format kann dasselbe LOD gefordert werden.

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	highest
Absolute 3D point accuracy (position/height)	lower than LOD1	$5/5m$	$2/2m$	$0.5/0.5m$	$0.2/0.2m$
Generalisation	maximal generalisation	object blocks as generalised features; $> 6 \cdot 6m/3m$	objects as generalised features; $> 4 \cdot 4m/2m$	object as real features; $> 2 \cdot 2m/1m$	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalised objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6 m	prototypes, higher 2 m	prototypes, real object form
PlantCover	no	$> 50 \cdot 50m$	$> 5 \cdot 5m$	$< LOD2$	$< LOD2$
... to be continued for the other feature themes					

Tabelle 7 LOD0-4 des CityGML 2.0 Standards und ihre Genauigkeitsanforderungen (Gröger et al., 2012; Albert et al., 2003).

Modellierungskonvention

Um eine einheitliche Verarbeitung von CityGML Eingangsdaten realisieren zu können, muss sich auf eine Modellierungskonvention für Geometrie und Semantik verlassen werden können. Es kann sich dabei auf die Handbücher für die Modellierung von 3D-Objekten der SIG3D (Special Interest Group 3D) gestützt werden (`http://en.wiki.quality.sig3d.org/index.php/Modeling`). Für Modelle in CityGML 3.0 könnte sich darauf verlassen werden, dass diese Konformität zum Dokument *CityGML 3.0 - Spaces and their allowed space boundaries.xlsx* (`https://docs.google.com/spreadsheets/d/1UtWiH7qVXClpq_rFHMM5P_zLmp5e-li1_ECi1amK1p4/edit#gid=772795371`) aufweisen. Modelle, die Abweichungen zu den Modellierungskonventionen, oder ungültige Geometrien aufweisen, können zu Fehlern in der weiteren Verarbeitung führen. Zusätzlich zu den textlichen Modellierungskonventionen könnte ein UML-Profil für die Lageplan ADE erstellt werden, das die Modellierung der Gebäude auf die gewünschte Art und Weise über OCL ”erzwingt”.

Geometrische Konsistenz

Im Falle eines Anbaus zu einem bestehenden Gebäude kann es zu geometrischen Inkonsistenzen kommen. Das liegt daran, dass im Zuge des Bauvorhabens die Koordinatenlage des architektonischen Modells in der Regel eine höhere Genauigkeit aufweisen wird, als die Daten, welche von den Vermessungsämtern bezogen werden. Würde das existierende Gebäude als LOD2-CityGML-Gebäudemodell hinzugeladen (dieses wird aus ALKIS-Gebäudegrundrissen, Laserscanning-Daten, Gebäudeeinmessungen oder Luftbild-basierten digitalen Oberflächenmodellen abgeleitet und weist damit im Zweifel eine geringere Genauigkeit in den Koordinaten auf, als das architektonische Modell), kann es sein, dass Lücken, Überschneidungen oder anderweitige Unstimmigkeiten zwischen den Geometrien des Bestands und des Architektenmodells zum Vorschein kommen, da die Koordinatengenauigkeit verschieden ist. Um diese Inkonsistenzen zu verhindern, wird bei der Erstellung des Lageplans der Bestand von den zuständigen Vermessungsingenieur*innen nochmals in hoher Genauigkeit aufgenommen. Der Gebäudebestand dient dem*der Architekt*in anschließend als Grundlage der Planung des Gebäudeanbaus. Für die Einbettung in den 3D-Lageplan, sowie die Abstandsflächenberechnung ist erforderlich, das sowohl das Architektenmodell des Anbaus, als auch der neu-aufgenommene Gebäudebestand als 3D-Modell vorliegen.

Gebäudeklasse und Vollgeschossanzahl

Die Gebäudeklasse (*class*-Attribut), sowie die Anzahl der überirdischen Vollgeschosse (*storeysAboveGround*-Attribut) des geplanten Bauwerks müssen im Datensatz des CityGML-Gebäudemodells verpflichtend bereitgestellt werden.

Relevante Flächen von Gebäudekomponenten für die Abstandsflächenberechnung

Gebäudekomponenten, die Abstandsflächen werfen können, sind den thematischen Gruppen Wandflächen (inklusive Gesimse), Dachflächen, sowie Gebäudeinstallationen (wie z.B. Vorbauten), welche wiederum aus Wand- und Dachflächen bestehen können, zuzuordnen.
Die für die Modellierung dieser Gebäudekomponenten verwendeten CityGML-Basisklassen sind:

- GroundSurface
- WallSurface
- RoofSurface
- OuterCeilingSurface
- OuterFloorSurface
- ClosureSurface

(Gröger et al., 2012; Kolbe et al., 2020)

Die CityGML-Klasse *BuildingInstallation* muss gesondert betrachtet werden, da Instanzen dieser Klasse aus den vorher aufgelisteten Basisklassen bestehen können. Dies muss jedoch nicht zwangsläufig der Fall sein, da *BuildingInstallation*-Objekte auch mit einfacher Geometrie ohne Semantik instantiiert werden können (Gröger et al., 2012; Kolbe et al., 2020).

Transformation des Gebäudemodells in das Landessystem

Für kleinräumige Projekte unter 1 km kann eine 2D-Helmerttransformation für die Überführung des Gebäudemodells vom lokalen kartesischen Projektkoordinatensystem in das UTM-Landessystem verwendet werden, da die Verzerrung nahezu einer Konstanten gleicht (Mitchell, 2020). Über mindestens zwei in der Lage korrespondierende Punkte im Projektkoordinatensystem und dem Landessystem können die zwei Translationsparameter t_x, t_y, der Rotationswinkel um den Ursprung ϵ, und der Skalierungsfaktor m bestimmt werden (Mitchell, 2020). Um die Höhen des Projekts (z-Komponente) in das Landessystem zu überführen, wird die Höhe des Höhenbezugssystems H in einem Punkt (z.B. des Urpsrungs des Pojektkoordinatensystems) des Projektes benötigt. Mittels dieser "Anbindungshöhe" und der lokalen Höhe z aus dem Projektkoordinatensystem können die Höhen der Projektpunkte im globalen Höhenbezugssystem abgeleitet werden (Mitchell, 2020).
Für große Projektgebiete, welche sich über mehrere Kilometer erstrecken, sind verschiedene Skalierungsfaktoren bzgl. der Streckenverzerrung für die verschiedenen Projektbereiche-/abschnitte zu ermitteln (Mitchell, 2020; Heunecke, 2017; Wasmeier, 2018).

Ankerpunkt und Rotationswinkel

Da das Gebäudemodell zumeist in lokalen kartesischen Koordinatensystemen vorliegt, muss ein sogenannter Ankerpunk und Rotationswinkel für die Überführung des Modells in das Landessystem bereitgestellt werden. Dies soll über eine Anlehnung an das von der Firma *Esri* entwickelte *World-File*-Format geschehen, welches Maßstabsfaktor, Translation, sowie Rotationswinkel für eine Koordinatentransformation in das Landessystem der Form

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & 0 & C \\ D & E & 0 & F \\ 0 & 0 & 1 & G \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.2}$$

beinhaltet. Diese Form der Transformation besteht aus einer 2D-Helmert-Transformation mit den Parametern *A, B, C, D, E, F*, welche die Koordinaten des Gebäudemodells in Lage und Maßstab überführt (Maßstab und Translation in Lage, Rotation um vertikale Achse), sowie einer von der Lage losgelösten und unabhängigen Translation der z-Komponente in das entsprechende Höhensystem über den Parameter *G*.

Das *World-File*-Format besteht aus sieben Zeilen, mit den sieben Parametern:

```
1. Zeile: A
2. Zeile: B
3. Zeile: C
4. Zeile: D
5. Zeile: E
6. Zeile: F
7. Zeile: G
```

Eine Beispiel-Datei *beispielWorldFile.wld* könnte wie folgt aussehen:

```
0.5000000000000000
0.8660254037844386
714268.3
−0.8660254037844386
0.5000000000000000
5322522.7
545.85
```

Digitales Geländemodell

Datenherkunft

Die Daten des DGM der näheren Umgebung des Bauvorhabens wird im Rahmen einer Bestandsvermessungen in hoher Auflösung per Laser-Scanning oder Photogrammetrie (z.B. über UAV (Unmanned Aerial Vehicle)) gewonnen. Die DGM-Gitter, welche von den Landesvermessungsämtern bereitgestellt werden liegen mit ihrer Auflösung im 1- bis 5-Meter-Bereich. Diese Auflösung ist für die Berechnung von Abstandsflächen, trotz der Möglichkeit, zwischen den Gitterpunkten zu interpolieren, nicht ausreichend. Bspw. kann im Falle von Grenz-naher Bebauung ein Wandteil, welches mit dessen vertikalen Begrenzungen nicht auf Gitterpunkten des verwendeten DGMs zu liegen kommt, in einer Abstandsfläche resultieren, die verbotenerweise die Nachbargrenze überschreitet.
Um die Auswirkung dT von kleinen Veränderungen $\mathbf{dx}$ in den Eingangsparametern bzgl. der resultierenden Abstandsflächentiefe abschätzen zu können, wird die partielle Ableitung von Gleichung 4.1 gebildet:

$$dT = \frac{\partial T}{\partial(DGM(P_0'), DGM(P_1'))} \cdot \mathbf{dx} = \frac{a}{2} \cdot \begin{bmatrix} -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \Delta DGM(P_0') \\ \\ \Delta DGM(P_1') \end{bmatrix} \tag{4.3}$$

Gleichung 4.3 zeigt, dass die Änderung der Abstandsflächentiefe dT eine lineare Abhängigkeit von der Summe der Komponenten des Verschiebungsvektors $\mathbf{dx}$ aufweist. In Abbildung 36 ist der lineare Verlauf der Änderung von dT für verschieden Faktoren a dargestellt. Durch das Bilden des Mittelwerts bei Geländeneigung, sowie der Multiplikation des Faktors a wird die Auswirkung von $\mathbf{dx}$ nicht 1:1 bemerkbar, sondern mit dem Faktor $\frac{a}{2}$ abgeschwächt.

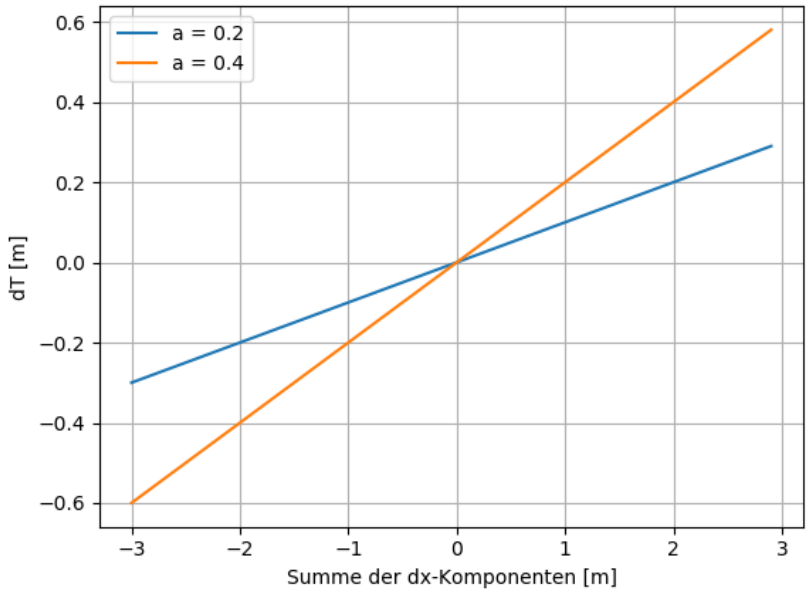


Abbildung 36 Lineare Änderung dT der Abstandsflächentiefe in Abhängigkeit der Summe der Komponenten des Verschiebungsvektors $\mathbf{dx}$.

In Abbildung 37 wird ein Szenario dargestellt, bei dem ein DGM-Gitter mit grober Auflösung für die Bestimmung der Abstandsfläche verwendet wird. Die Geländehöhe an den auf die x-y-Ebene projizierten, vertikalen Begrenzungen der Wand wird durch lineare Interpolation zwischen den zwei Gitterpunkten des DGMs ermittelt. Dadurch ergeben sich für die Positionen P_0' und P_1' die in gelb dargestellten Höhen h_0' und h_1'. Die "wahren" Höhen des Geländes sind jedoch die Variablen $\hat{h}_0$ und $\hat{h}_1$. Die Differenzen zwischen den Höhen $\hat{h}_0$ und h_0' bzw. $\hat{h}_1$ und h_1' werden als dh_0 bzw. dh_1 bezeichnet und in pink abgebildet.
Durch die interpolierten Höhen und den damit entstehenden Änderungen der Wandhöhen $h_0 + (h_0' - \hat{h}_0) = h_0 + dh_0$ und $h_1 + (h_1' - \hat{h}_1) = h_1 + dh_1$ wird die Abstandsflächentiefe um $-\frac{a}{2} \cdot (dh_0 + dh_1)$ verfälscht. Je nachdem, ob die Summe aus dh_0 und dh_1 negatives oder positives Vorzeichen besitzt, fällt die Abstandsflächentiefe zu groß oder zu klein aus, als sie eigentlich wäre. Aus diesen Gründen ist es v.a. in der Nähe von Nachbargrenzen notwendig, an der geplanten baulichen Anlage ein hoch-auflösendes DGM zu Verfügung zu haben, um die Abstandsflächen mit hoher Genauigkeit berechnen zu können und Rechtsstreit zu vermeiden.
Für die Peripherie des Bauvorhabens ist ein DGM mit Auflösung im Meter-Bereich ausreichend, da hier keine Berechnungen bzgl. der Abstandsflächen erforderlich sind.

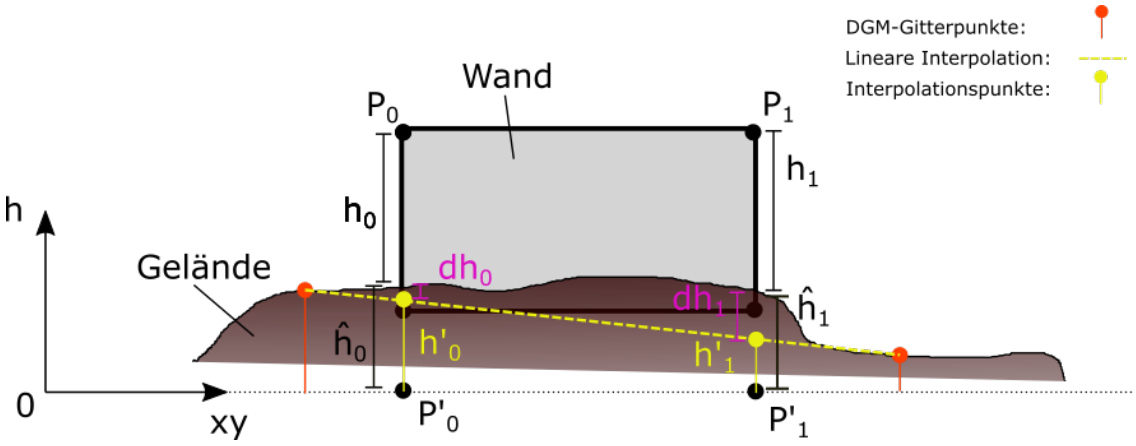


Abbildung 37 Darstellung der fehlerhaften Bestimmung der Abstandsflächentiefe bei Verwendung eines zu grob-auflösenden DGMs.

Koordinatenreferenzsystem der Eingangsdaten

Zielkoordinatensystem der Abstandsflächen

Da das Resultat der Abstandsflächenberechnung im (amtlichen) Lageplan dargestellt werden soll, ist zunächst festzustellen, in welchem CRS (Coordinate Reference System) der Lageplan erforderlich ist. Nach BauPrüfVO NRW §3 Absatz 1 mit Stand vom 20.8.2020 ist der Lageplan auf Grundlage des Katasterauszugs anzufertigen. Daraus folgt, dass der Lageplan dem ETRS89 (European Terrestrial Reference System 1989)/UTM-System für die Lage bzw. dem GCG2016 (German Combined QuasiGeoid 2016) als Höhenbezugsfläche zugrunde liegt, da die amtlichen Daten aus AFIS, ALKIS und ATKIS in diesem CRS bereitgestellt werden (Heunecke, 2017). Außerdem ist in der BauPrüfVO NRW §§3, 4 gefordert, sich bei Lage- und Höhenangaben auf das amtliche Lage- oder Höhenbezugssystem zu beziehen. Zumindest das Gebäudemodell muss also spätestens nach der Berechnung der Abstandsflächen in das ETRS89/UTM-System transformiert werden. Bzgl. der Abstandsflächen muss beachtet werden, dass sich die Handhabung von Flächen und Strecken im Liegenschaftskataster und (amtlichen) Lageplan unterscheiden. Die im Liegenschaftskataster ausgewiesenen Flächen beziehen sich auf die Fläche auf dem Bezugsellipsoid (*Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten*, 2019). Streckenmaße in (amtlichen) Lageplänen werden hingegen als Horizontalstrecken vor Ort angegeben (*Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten*, 2019).

Gemeinsames Koordinatenreferenzsystem

Um die Daten aus den verschiedenen Fachbereichen aus Vermessung und Planung für die Abstandsflächenberechnung zusammenzuführen, müssen die Datensätze in ein gemeinsames Koordinatenreferenzsystem (CRS) überführt werden. Das Gebäudemodell des*der Planer*in liegt meist in einem lokalen kartesischen Rechtssystem vor, wohingegen die Daten aus dem Bereich der Vermessung in der Regel mit absoluten Koordinaten im Landessystem ETRS89/UTM geo-referenziert sind (Heunecke, 2017; Brüggemann & von Both, 2015; Aumann et al., 2016).

Aufgrund dessen existieren zwei Möglichkeiten, in welchem CRS die Abstandsflächen berechnet werden können:

1) Die Abstandsflächenberechnung findet im UTM-Landessystem statt, das Gebäudemodell wird also direkt in das jeweilige CRS überführt.

2) Die geo-referenzierten Vermessungsdaten (DGM, Grundstücksgrenzen) werden in das lokale Rechtssystem des Gebäudemodells transformiert und die Berechnung der Abstandsflächen findet lokal statt. Anschließend werden Abstandsflächen und Gebäudemodell in das UTM-System transformiert.

Den unterschiedlichen Ansätzen geschuldet, muss untersucht werden, welche Auswirkungen es hat, wenn die Abstandsflächen in einem lokalen kartesischen Koordinatensystem (vor Ort) berechnet werden, oder wenn sie im UTM-Landessystem berechnet werden.

Abbildungsverzerrungen bei der UTM-Abbildung

Die UTM-Abbildung ist eine konforme Abbildung (d.h. Richtungswinkel bleiben nach der Projektion auf die Ebene erhalten) auf einen Zylinder (Aumann et al., 2016). Die Streifenbreite pro Zone beträgt 6°, die Schnittstreifen weisen eine Breite von 360 km auf (Aumann et al., 2016; AKG Software, 2019; Bayerische Vermessungsverwaltung, 2009). Die Zonen werden bzgl. festgelegter Haupt- oder Mittelmeridiane definiert (Aumann et al., 2016). Diese legen die gekrümmte Erdoberfläche fest, die auf eine Ebene projiziert wird (Bayerische Vermessungsverwaltung, 2009). Die Streifenbreite ist der Abstand zwischen den Schnittlinien des Zylinders mit dem Referenzellipsoid (Bayerische Vermessungsverwaltung, 2009).

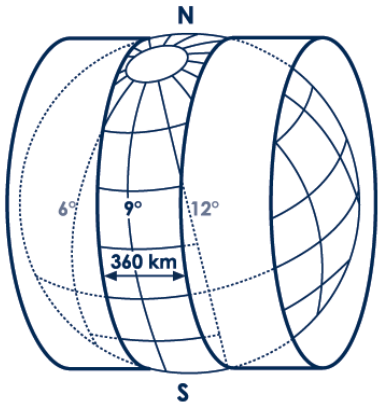


Abbildung 38 Darstellung der Zylinderabbildung bei der UTM-Projektion (Aumann et al., 2016).

Die Streckenverzerrung bei der Abbildung von örtlichen Strecken in das UTM-Abbildungssystem nimmt mit dem Abstand zum Hauptmeridian quadratisch zu (Aumann et al., 2016). In Abbildung 39 ist zu erkennen, dass die Maßstabsänderung am Hauptmeridian maximal ist. Mit Annäherung an die Schnittmeridiane nimmt die Verzerrung weiter, ab bis sie bei einem Abstand von ca. (circa) 180 km zu Null wird, und danach wieder ansteigt.

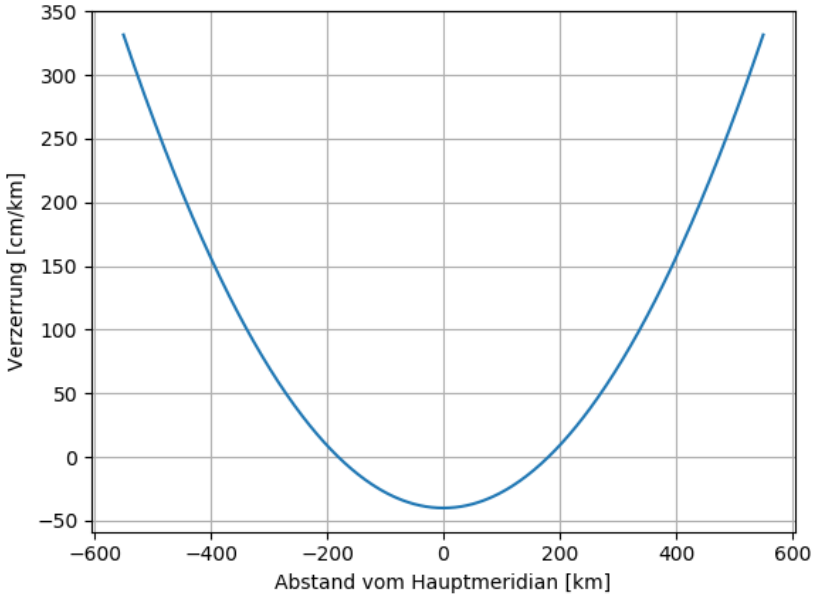


Abbildung 39 Verzerrung (bei Strecken auf dem Ellipsoid) pro Kilometer in Abhängigkeit des Abstands zum Hauptmeridian.

Aufgrund der Streckenverzerrung wird, analog zu den Flächengrößen, zwischen drei verschiedenen Arten von Strecken unterschieden (siehe Abbildung 40):

- *Strecke vor Ort*: Gemessene Horizontal-/Schrägstrecke zwischen zwei Punkten
- *Strecke auf Ellipsoid*: Strecke auf dem Ellipsoid; Strecke vor Ort kann über Höhenreduktion auf Ellipsoid überführt werden
- *Strecke in Abbildungsebene*: Aus Koordinaten der Projektionsebene berechenbar oder über Abbildungsreduktion der Strecke auf Ellipsoid

(*Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten*, 2019)

Abbildung 40 Visualisierung der verschiedenen Bezugssysteme der Strecken und deren Reduktionsbeziehungen untereinander (*Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten*, 2019).

Es ist darauf hinzuweisen, dass die jeweiligen Lagekoordinaten des Start- und Zielpunktes, auf die Projektionsebene abgebildet, für alle drei Strecken dieselben sind. Allein die Strecke zwischen den Punkten unterscheidet sich je nach Bezugssystem.
Aus den vorangegangenen Ausführungen folgt, dass Strecken, welche aus Koordinaten in der Projektionsebene berechnet werden, ein anderes Maß aufweisen, als Strecken vor Ort (siehe Abbildung 41). Strecken vor Ort werden über zwei Reduktionsschritte von der Örtlichkeit auf das Ellipsoid, und schließlich in die Projektionsebene überführt (Heunecke, 2017):

1) Streckenreduktion auf die Strecke S_{ell} auf Ellipsoid:

$$S_{ell} = S_H \cdot (1 - \frac{h_{ell}}{R_B}) = S_H \cdot m_H \tag{4.4}$$

, mit der horizontalen Strecke zwischen zwei Punkten vor Ort S_H, der mittleren ellipsoidischen Höhe $h_{ell} = \frac{1}{2} \cdot (h_1 + h_2)$ der beiden Punkte, sowie dem Radius der Gauß'schen Schmiegekuge $R_B = \sqrt{M_B \cdot N_B}$, wobei $N_B = \frac{a}{(1-e^2 sin^2 B)^{\frac{1}{2}}}$ und $M_B = \frac{a(1-e^2)}{(1-e^2 sin^2 B)^{\frac{3}{2}}}$. a, b, e sind die das Referenzellipsoid definierenden Parameter der großen und kleinen Halbachse, sowie der ersten numerischen Exzentrizität.

2) Abbildungsreduktion auf die Strecke S_{UTM} auf Abbildungsebene:

$$S_{UTM} = S_{ell} \cdot m_0 \cdot (1 + \frac{y_1^2 + y_1 y_2 + y_2^2}{6 m_0^2 R_B^2}) = S_{ell} \cdot m_{UTM} \approx S_{ell} \cdot m_0 \cdot (1 + \frac{y_m^2}{2 m_0^2 R_B^2}) \tag{4.5}$$

, mit dem Abbildungsmaßstab $m_0 = 0.9996$, den von Zonenziffer und Offset (-500000 [m]]) befreiten Ostwerten y_1, y_2 bzw. dem mittleren Ostwert $y_m = 0.5(y_1 + y_2)$ und der Strecke auf dem Ellipsoid S_{ell}.

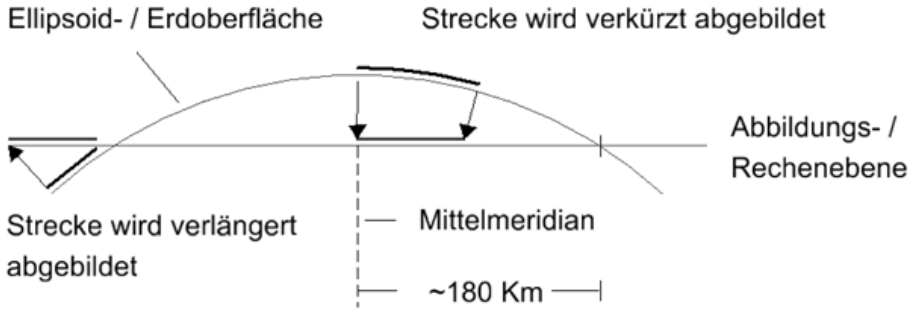


Abbildung 41 Streckenverzerrung bei der Abbildung der Strecken von Ellipsoid/Geländeoberfläche auf die Projektionsebene der UTM-Projektion (Kaden, 2016).

Folglich ist festzustellen, dass das Maß einer Strecke zwischen zwei Punkten, welche in einem lokalen System gemessen wurden, und das Maß derselben Strecke projiziert auf die UTM-Abbildungsebene nicht identisch sind. Im nächsten Schritt ist zu klären, ob, und wenn ja, welche Auswirkungen das auf die Berechnung der Abstandsflächen hat.

Auswirkungen auf die Maße der Abstandsflächen

Wie in Gleichung 4.1 ersichtlich, gehen in die Berechnung der Abstandsflächentiefe T keinerlei Strecken (z.B. des Gebäudes) ein. D.h., für die Berechnung der Abstandsflächentiefe an sich, ist es völlig unerheblich, in welchem CRS diese durchgeführt wird, da nur relative Höhenunterschiede zwischen Gelände und Gebäudekomponenten relevant sind.
Allerdings ist die Abstandsfläche als geometrisches Objekt ein Polygon mit linearen Streckenstücken zwischen den Polygonpunkten. Nach BauO NRW §6 Absatz 4 mit Stand vom 20.8.2020 wird definiert, dass die Abstandsflächentiefe senkrecht zur Außenwand gemessen wird. Für die Berechnung der Punkte des Abstandsflächenpolygons gehen neben der Abstandsflächentiefe T, die Lagekoordinaten der vertikalen Begrenzungen des Wandteils ein. Die Lagekoordinaten der vertikalen Begrenzungen werden benötigt, um die Richtungsvektoren und Längen (als Faktor für die Richtungsvektoren) zu ermitteln, welche für die Berechnung der restlichen Knotenpunkte des Polygons erforderlich sind (siehe Abbildung 42).

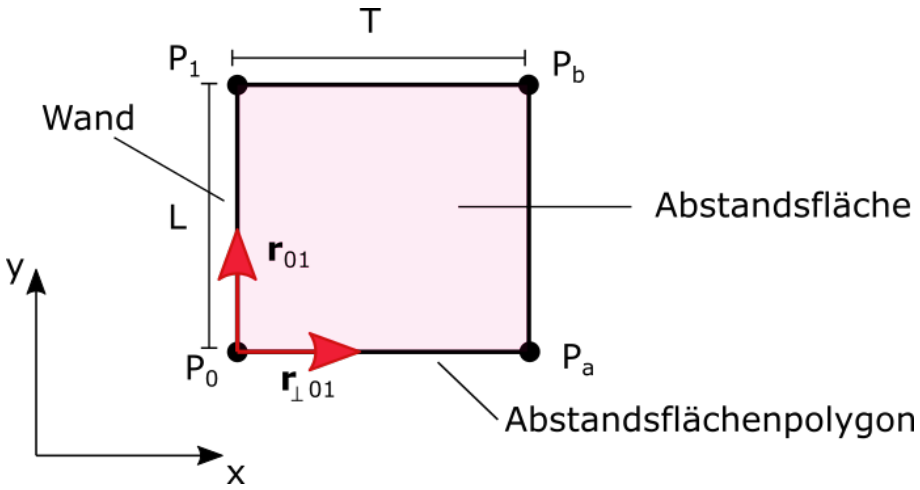


Abbildung 42 Über die Lagekoordinaten der vertikalen Wandbegrenzungen P_0 und P_1 werden normierte Richtungsvektoren $\mathbf{r}_{01}$ und senkrecht dazu $\mathbf{r}_{\perp 01}$ berechnet. Diese werden mit den Faktoren der Wandlänge L bzw. der Abstandsflächentiefe T skaliert. Mittels der Geradengleichung $P_{neu} = P_{Aufpunkt} + \lambda \cdot \mathbf{r}_{Richtung}$ werden dann die Knotenpunkte des Abstandsflächenpolygons berechnet.

Aus den vorigen Ausführungen folgt, dass sich die Abstandsflächenpolygone vom UTM-System zu deren realer Absteckung vor Ort unterscheiden werden. Die relativen Beziehungen bleiben durch die festgelegten Reduktionsvorschriften von einem System ins andere erhalten und nachvollziehbar. Problematisch wird es, wenn gefordert wird, dass die Abstandsflächentiefe vor Ort der anhand des Gebäudemodells berechneten Abstandsflächentiefe entsprechen soll. Es macht einen Unterschied, ob die Abstandsflächen im UTM-System berechnet wurden, denn dann wird die örtlich abgesteckte Abstandsflächentiefe von der theoretisch berechneten Tiefe verschieden sein. Besonders wichtig ist die genaue Lage der Abstandsflächen bzgl. von Grenzen.

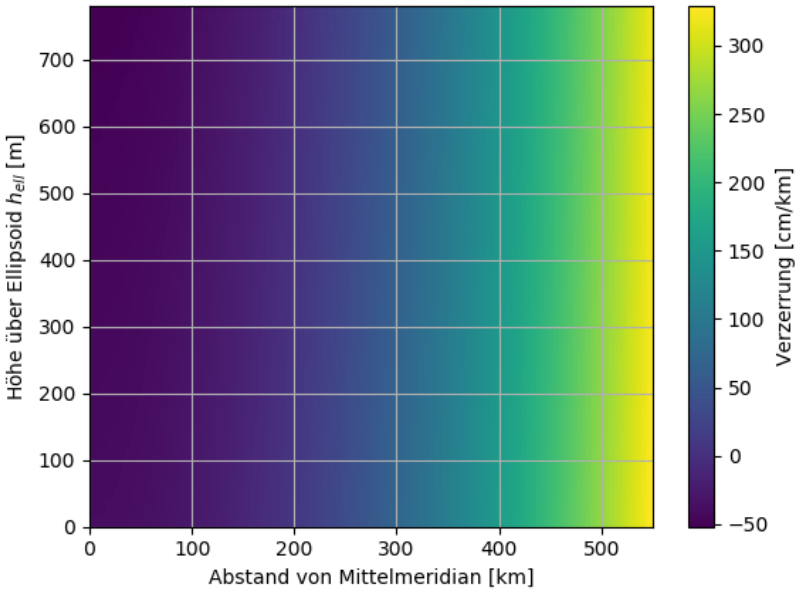


Abbildung 43 Differenz zwischen der örtlichen Strecke $S_H = 10\,m$ und der projizierten Strecke im UTM-System in Abhängigkeit des Meridianabstands und der ellipsoidischen Höhe mit mittlerem Krümmungshalbmesser $R = 6381\,km$ für mittlere Breiten Deutschlands.

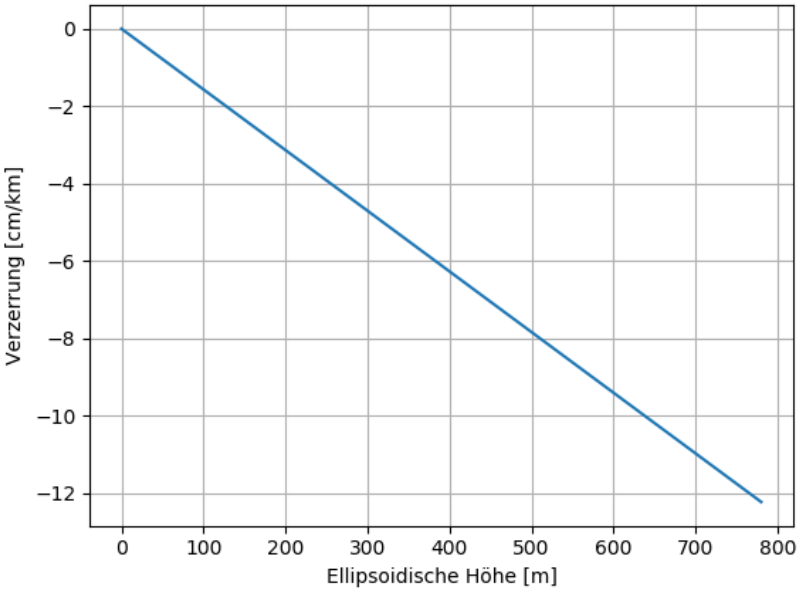


Abbildung 44 Verzerrung der Strecke bei der Reduktion von der örtlichen Strecke mit ellipsoidischen Höhen auf das Referenzellipsoid.

Die Grundstücksgrenzen und Gebäudeeinmessungen für das Liegenschaftskataster werden in der Regel mit Zentimeter-Genauigkeit erfasst (*Eigentumsgrenzen im Liegenschaftskataster - Hinweise für Grundstückseigentümer*, 2017; *Gebäudeeinmessung zur Sicherung des Eigentums an Grund und Boden*, o. J.; *Grenzfeststellung - Grenzwiederherstellung oder Grenzermittlung?*, o. J.; Noack et al., 2005). Da sich die Differenzen der Strecken von der Örtlichkeit zum UTM-System sich auch im unteren Zentimeter-Bereich bewegen (siehe Abbildung 43), wird davon ausgegangen, dass für Abstandsflächen, die keine extremen Maße in ihrer Abstandsflächentiefe aufweisen und sich deren Bauprojekt nicht zu weit vom Mittelmeridian der entsprechenden UTM-Zone befindet, die Rolle des Bezugssystems, in welchem diese berechnet wird, eine untergeordnete Rolle spielt. Auch haben die ellipsoidischen Höhen bei kurzen Strecken von 10 m keine großen Einfluss (im Millimeterbereich bei ellipsoidischen Höhen von 800 m) auf die Streckenreduktion (siehe Abbildung 44).
Dies bestätigt eine grobe Abschätzung für Breitengrade von $\varphi = 51°$ mit Krümmungsradius der Schmiegekugel $R = 6381\,km$ (nach Aumann et al. (2016)), sowie Zonenbreiten von $\Delta\lambda = 6°$. Der maximalen Abstand zu einem Mittelmeridian ergibt sich über $s = 0.5 \cdot R \cdot \sin(\frac{\pi}{2} - \varphi) = 0.5 \cdot R' \cdot \Delta\lambda \approx 210\,km$. Wird mit Abbildung 43 die Streckendifferenz für Strecken (hier Abstandsflächentiefe) von 10 m für den Meridianabstand von 210 km ermittelt, so bewegen sich die Werte im Bereich von ca. 0.5 cm bis 1 cm.

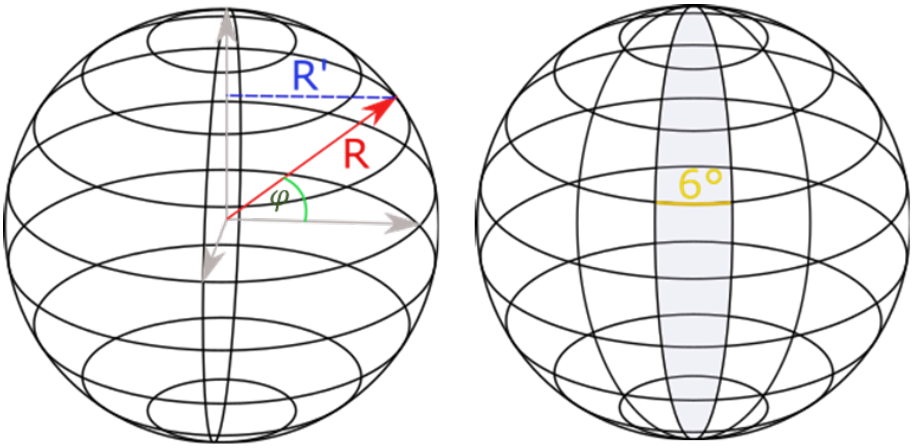


Abbildung 45 Darstellung der Größen für die Abschätzung des maximalen Abstands zum Hauptmeridian für einen bestimmten Breitengrad.

Die vorigen Überlegungen, und die Tatsache, dass das DGM, sowie die Grundstücksgrenzen im Landessystem vorliegen, führen zu dem Entschluss, zunächst das Gebäudemodell in das Landessystem zu überführen und anschließend in diesem die Abstandsflächenberechnung durchzuführen.

Konfigurationsdatei für Parameter und Berechnungsregeln

Um die Abstandsflächen berechnen zu können, ist der Algorithmus in mathematischer Form notwendig. Da sich die Berechnungsregeln von Bundesland zu Bundesland und deren BauO unterscheiden können, ist es sinnvoll in dem Programm eine Möglichkeit zu integrieren, mit der sich die Berechnungsregeln zu Laufzeit über eine bereitgestellte Konfigurationsdatei anpassen lassen. Dies könnte sich auf die Wahl der Werte von Parametern beschränken, aber auch die mathematischen Formulierungen mit einschließen.

Dynamische Evaluierung des mathematischen Ausdrucks zur Laufzeit

Viele Programmiersprachen wie u.a. Java oder Python bieten die Möglichkeit, Zeichenketten, welche Quellcode in der jeweiligen Sprache enthalten, zur Laufzeit auszuführen oder sogar Definitionen von Funktionen/Methoden zu kompilieren. Damit können mathematische Ausdrücke und Parameter bzgl. der Abstandsflächenberechnung über eine Konfigurationsdatei zur Laufzeit in das Programm geladen und ausgewertet werden. Dies ermöglicht einen flexible Anpassung der verschiedenen Berechnungsregeln und Parameter, die sich von Bundesland zu Bundesland unterscheiden können. Die mathematischen Ausdrücke und Parameter können einfach als strukturierte Text-Datei bereitgestellt werden.
Zur Vereinheitlichung muss ein konsistentes Datenformat für die Zurverfügungstellung der Information gefunden werden, welches von der Software verarbeitet werden soll. Dafür bieten sich gängige Formate wie INI, YAML, TOML, XML oder JSON an (Sananthana, 2019; Ueding, 2017). Auch muss Inhalt und Struktur der Datei definiert werden um Mehrdeutigkeiten zu vermeiden. Daher ist ein Datenformat zu bevorzugen, dass es erlaubt Datentypen, Existenz von Elementen, usw. über ein Schema festzulegen. Hierfür kommt neben XML auch JSON in Frage.

Grundstücksgrenzen

Grundstücksgrenzen des zu bebauenden Eigentums, sowie der benachbarten Grundstücke werden als (2D-)Polygon-Objekte aus dem Katasterauszug entnommen. Sie wird für die Privilegierung einzelner Gebäudekomponenten benötigt, die eine gewisse Lage zur Grundstücksgrenze aufweisen.

Flächen aus dem Bebauungsplan

Aus dem Bebauungsplan werden das Art der baulichen Nutzung des Baugebiets des geplanten Gebäudes, sowie angrenzende öffentliche Flächen benötigt. Daraus ergibt sich nach BauO NRW §6 Absatz 5 mit Stand vom 10.9.2020 ein Faktor a, mit dem die berechneten Außenwandhöhen H_i multipliziert werden und sich die Abstandsflächentiefen T_i ergeben. Auch hier werden, wie schon im Abschnitt der Grundstücksgrenzen, die Flächen als 2D-Polygon-Objekte erwartet.

Eingangsdaten-Definition durch CityGML-Profile

Wird gewünscht, dass eine Software die automatische Abstandsflächenberechnung anhand eines einzelnen Datensatzes durchführt, sind die Eingangsdaten in einem globalen Datenomdell zusammenzufassen. Der Vorteil dieses Ansatzes wäre, dass die Daten schon von vornherein in einem einheitlichen Datenmodell vorliegen und sich bei der Software-Entwicklung auf die Berechnung des Abstandsflächen fokussiert werden könnte. Sind die Eingangsdaten jedoch auf verschiedene Datensätze und Datenmodelle bzw. Datenformate verteilt, bedeutet dies, dass die unterschiedlichen Datensätze erst in ein gemeinsames Datenmodell für die weitere Verarbeitung transformiert werden müssten.
Für die Berechnung am Gebäude selbst wurde bereits CityGML als das am besten geeignete Datenmodell identifiziert. Im CityGML 2.0-Standard würden für die weiteren Eingangsdaten die Module *LandUse* und *Relief* bereitstehen (Gröger et al., 2012).
Für CityGML 3.0 wird im Zuge dieser Arbeit ein Datenmodell zur Modellierung der Lageplan-Objekte entwickelt, sodass die Klassen für die Abbildung des Katasters und der räumlichen Planung auch für CityGML 3.0 vorhanden wären. Es liegt daher nahe, dass die Wahl für ein globales Datenmodell, das alle Eingangsdaten vereint, auf den CityGML-Standard fällt.
In Tabelle 8 ist die Abbildung der Eingangsdaten auf CityGML-Module dargestellt.

Eingangsdaten	**Abbildung auf CityGML-Module**	
	CityGML 2.0	***CityGML 3.0***
Gebäudmodell	*Building*-Modul	*Building*-Modul
DGM	*Relief*-Modul	*Relief*-Modul
Flur-/Grundstück	*LandUse*-Modul; Spezifizierung durch Code-Listen	*LandAdministrationCadastre*-Paket der SiteplanADE
Flächen aus dem Bebauungsplan	*LandUse*-Modul; Spezifizierung durch Code-Listen	*LandUsePlanning*-Paket der SiteplanADE

Tabelle 8 Abbildungstabelle der Eingangsdaten auf Module von CityGML2.0 und CityGML3.0.

Eingangsdatenmodell für CityGML 2.0

Die benötigten CityGML 2.0-Module für die Abbildung aller benötigten Daten für die Abstandsflächenberechung sind zum einen das *Building*-Modul, welches für das zu inspizierende Gebäudemodell verwendet wird. Zum anderen wird das *Relief*-Modul benötigt, welches das das Gebäude umgebende Terrain abbilden soll. Grundstücksgrenzen und Flächen aus dem Bebauungsplan werden gleichermaßen mittels der Klasse *LandUse* aus dem gleichnamigen Modul modelliert. Die Spezifizierung der *LandUse*-Objekte in Grundstück oder Art der Fläche des Bebauungsplans muss über Code-Listen geschehen.

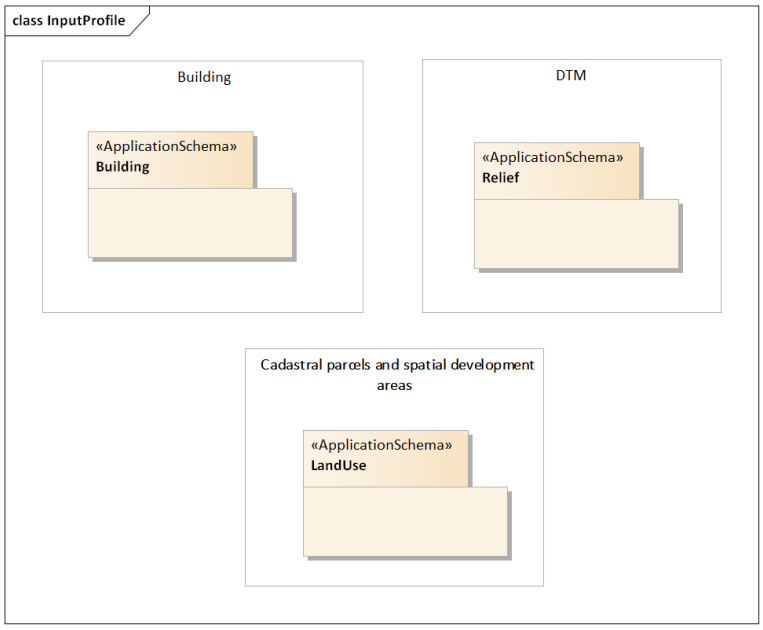


Abbildung 46 Module für das Eingangsdatenprofil für CityGML 2.0.

Eingangsdatenmodell für CityGML 3.0

Durch die SiteplanADE sind alle Klassen enthalten, die für einen vollständigen Datensatz bzgl. der Abstandsflächenberechnung erforderlich sind. Dadurch, dass die Klassen hier explizit modelliert bereitstehen, kann das Profil sogar auf Klassen-Ebene definiert werden, im Gegensatz zu dem Profil für CityGML 2.0. Identisch zum Profil für CityGML 2.0 werden die Module *Building*, *Relief* für die Modellierung von Gebäude bzw. Gelände verwendet. Für die Abbildung von Kataster-Parzellen steht aus dem SiteplanADE-Paket *LandAdministration* die Klasse *CadastralParcel* bzw. *AX_Flurstueck* zur Verfügung. Die Flächen aus dem Bebauungsplan sind durch Klassen aus dem SiteplanADE-Paket *LandUsePlanning* abgedeckt.

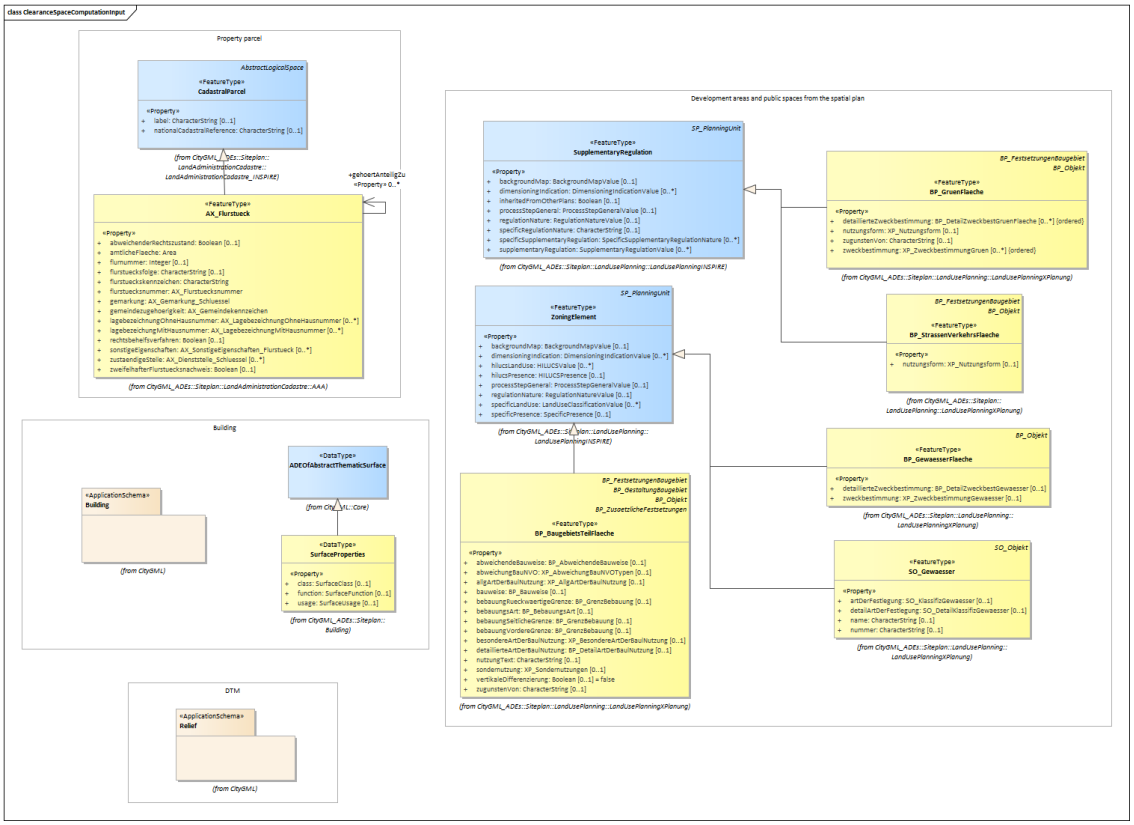


Abbildung 47 Module für das Eingangsdatenprofil für CityGML 3.0.

Bedeutung für die Software-Entwicklung

Für die Software-Entwicklung bedeutet die vorherige Definition von CityGML-Profilen, dass für jede individuelle Software zusätzlich Software-spezifische ADEs und OCL-Konditionen bzgl. der CityGML-Profile erzeugt werden müssen. Die ADEs und OCL-Bedingungen definieren sich je nachdem, wie die Datenextraktion aus dem Datensatz erfolgt, welche Objekt-Klassen zwingend für die Berechnung der Abstandsflächen erforderlich sind, welche Objekt-Klassen ggf. mittels einer ADE erstellt werden müssen, und welche Objekt-Attribute notwendig sind, ausgeschlossen, oder zusätzlich über eine ADE hinzugefügt werden sollen.

4.2 Implementierung der automatisierten Abstandsflächenberechnung

4.2.1 Einführung zum Algorithmus der automatisierten Abstandsflächenberechnung

Laufzeitumgebung des Algorithmus

Die spätere Verwendung des hier entwickelten Algorithmus für die automatisierte Abstandsflächenberechnung zielt auf die Einbettung in die Laufzeitumgebung der Safe Software FME ab. Daher erwartet der Programmteil für die automatisierte Abstandsflächenberechnung nur das Gebäudemodell im CityGML-Format. Die restlichen Daten wie DGM, Baugebietsflächen und Grundstücksgrenze werden dem Programmteil in FME-interner Objekt-Repräsentation übergeben und weiterverarbeitet.

Implementierung der Aktivitätsdiagramme

Die Aktivitätsdiagramme dienen der Plattform-neutralen Darstellung des Algorithmus der Abstandsflächenberechnung. Die Implementierung der Aktivitätsdiagramme erfolgt in dieser Arbeit mittels der Programmiersprache Python. Durch die Programmierung in Python wird es möglich, den Quellcode in die Safe Software FME über den Transformer *PythonCaller* einzubetten.
Jeder der Prozesse und Unterprozesse (dargestellt als blaue Rechtecke) korrespondiert mit einem Block von Quellcode, welcher den entsprechenden Prozess realisiert/implementiert. Der mit einem Prozess korrespondierende Quellcode verarbeitet die jeweiligen Eingangsobjekte bzw. gibt die Ausgangsobjekte aus (hell-gelbe/grüne Rechtecke). Die sequentielle Strukturierung des Quellcodes folgt der Navigation der Prozesse über die Pfeilrichtungen zum nächsten Prozess.

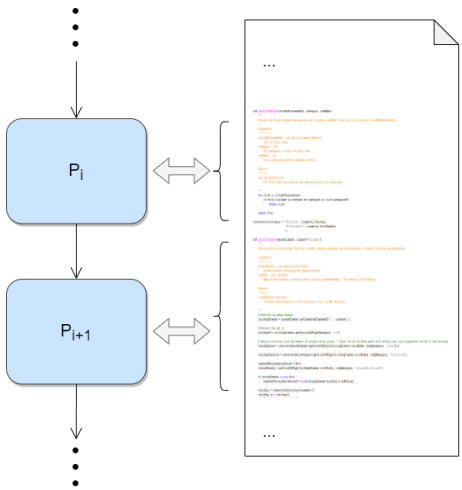


Abbildung 48 Die Prozesse P und ihre sequentielle Abfolge wird in Blöcken von Quellcode implementiert.

Überblick über den prinzipiellen Ablauf des Algorithmus

Im Aktivitätsdiagramm in Abbildung 49 wird die prinzipielle Sequenz der Arbeitsschritte des Algorithmus und die dafür benötigten Parameter/Daten dargestellt. Auf höchster Ebene lässt sich der Gesamtprozess in zwei Teile gliedern. Im ersten Teil werden die Eingangsdaten eingelesen und derart verarbeitet, dass sie für den Abstandsflächenalgorithmus applikabel sind. Erst im zweiten Teil finden dann die Regeln der Abstandsflächenberechnung Anwendung auf das Gebäudemodell.
Die Eingangsdaten bestehen aus einem validen CityGML-Gebäudemodell, Regeln und Parameter zur Berechnung der Abstandsflächen, dem DGM, sowie den Grundstücksgrenzen und Flächen aus dem geltenden Bebauungsplan.
Im Prozess *Vorverarbeitung* wird das eingehende CityGML-Modell mit Hilfe der Transformationsparameter in das Zielreferenzsystem überführt und auf eine Programm-internen Darstellung abgebildet (siehe Unterabschnitt 4.2.3). Die Grundstücksgrenze und die Flächen des Bebauungsplans werden ebenfalls in das Programm-interne Datenmodell überführt.
Zudem wird das DGM für die Weiterverarbeitung reduziert und nur Datenpunkte im näheren Umfeld des Gebäudes beibehalten. Mit Hilfe des DGMs werden die Schnittlinien der Gebäudeflächen mit dem Gelände, die sogenannten TICs (Terrain Intersection Curves), berechnet. Das Ergebnis der Vorverarbeitung geht zusammen mit den Flächen des Bebauungsplans, der Grundstücksgrenzen, sowie der Berechnungsregeln in den Prozess *Untersuchung auf Abstandsflächen* ein.
In den folgenden Abschnitten wird im Detail erläutert werden, welche Unterprozesse in den Hauptprozessen *Vorverarbeitung* und *Untersuchung auf Abstandsflächen* stattfinden.

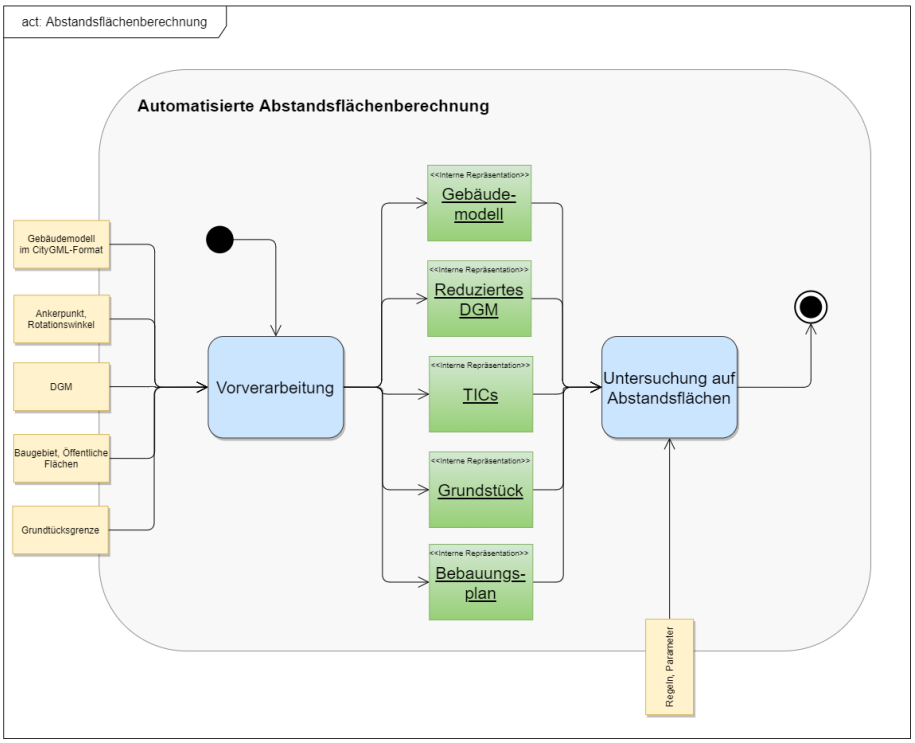


Abbildung 49 Aktivitätsdiagramm der automatisierten Abstandsflächenberechnung.

4.2.2 CityGML-Gebäude-Profile für die automatisierte Abstandsflächenberechnung

Gebäudeprofile für die automatisierte Abstandsflächenberechnung

Zunächst wird ein Profil für das Eingangsgebäudemodell festgelegt, um sicherzustellen, dass der Algorithmus das eingehende CityGML-Gebäudemodell verarbeiten kann. In diesem Fall besteht das CityGML-Profil für die Implementierung der automatisierten Abstandsflächenberechnung nur aus dem *Building*-Modul. Um sicherzustellen, dass die Eingangsdaten konform zur Funktionsweise des hier implementierten Algorithmus ist, werden weitere Einschränkungen der Gebäudeprofile für die CityGML-Versionen 2.0 und 3.0 definiert.

Gebäude-Profil für CityGML 2.0

UML-Modellierung

Das standardmäßige CityGML 2.0-Modul für die Repräsentation von Gebäuden ist für den Anwendungsfall der Abstandsflächenberechnung noch nicht ausreichend modelliert, weshalb die Entwicklung einer ADE notwendig wird.
Konkret fehlen Klassen, die Beziehungen zwischen Gebäudekomponenten abbilden können. Das ist notwendig, um bspw. die Adjazenz von Bauteilen (z.B. Gesimse, Balkone, etc.) zu ihrer übergeordneten Gebäudekomponente (z.B. Außenwand, Fassade, etc.) herzustellen. In der Version 3.0 des CityGML-Standards ist das über die Klasse *CityObjectRelation* möglich, sodass diese Klasse einfach im Profil für CityGML 2.0-Gebäude (welches über eine ADE realisiert wird) nachmodelliert werden kann.
Eine weitere Aspekt welcher die Erstellung des Gebäude-Profils notwendig macht ist, dass es nicht möglich ist, tiefergehende Information über die Rolle der Begrenzungsflächen (Subklassen der Superklasse *_BoundarySurface*) bereitzustellen. Für die Berechnung der Abstandsflächen ist es jedoch notwendig, z.B. relevante Fassaden zu identifizieren, oder im konkreten Fall von privilegierten Gebäudeinstallationen und Vorbauten deren Seitenwände. Um diese Informationen in den Begrenzungsflächen zu integrieren, werden in der ADE *class-, function-* und *usage-*Attribute für alle Unterklassen von *_BoundarySurface* eingeführt, deren Werte in externen Code-Listen definiert werden.
Um die Informationen über Klasse, Funktion und Nutzung der Begrenzungsflächen auch für Gebäudeinstallationen (in CityGML: *BuildingInstallation*) zugänglich zu machen, wird die neue Unterklasse *GenericThematicSurface* der Klasse *_BoundarySurface* in Anlehnung an die gleichnamige Klasse im CityGML 3.0-Standard erzeugt. Diese Klasse soll die Verwendung von reiner Geometrie ohne Semantik bei der Modellierung von Gebäudeinstallationen (*BuildingInstallation*) ersetzen und gleichzeitig notwendige Informationen der Begrenzungsflächen (wie z.B. Seitenwände) zugänglich machen.
Die zusätzlichen Attribute *class, function, usage* werden als Unterklassen von *_GenericApplicationPropertyOfBoundarySurface* modelliert. Dafür wird für die Ableitung eines XML-Schemas mit *ShapeChange* eine Unterklasse von *_BoundarySurface* erzeugt, die den Stereotype *«ADE-Element»* besitzt.

Die Klasse *GenericThematicSurface* ist eine einfache Unterklasse von *_BoundarySurface* ohne zusätzliche Attribute.
Auch die Objekt-Relation-Klasse *CityObjectRelation* wird als Unterklasse des generischen Attributs der Klasse *_CityObject* modelliert. Die Unterklasse *CityObject* trägt den Stereotype *«ADEElement»* und besitzt als Attribut die *CityObjectRelation*-Klasse, welche wiederum eine Instanz der Klasse *_CityObject* assoziiert. Dies entspricht einer Nachbildung der Assoziationsklasse *CityObjectRelation* aus dem CityGML 3.0-Modell, mit dem Unterschied, dass hier Objektrelationen nur über Referenzen zu Objekten realisiert werden sollen.
Die entsprechende XSD-Schema-Datei ist in Anhang A zu finden.

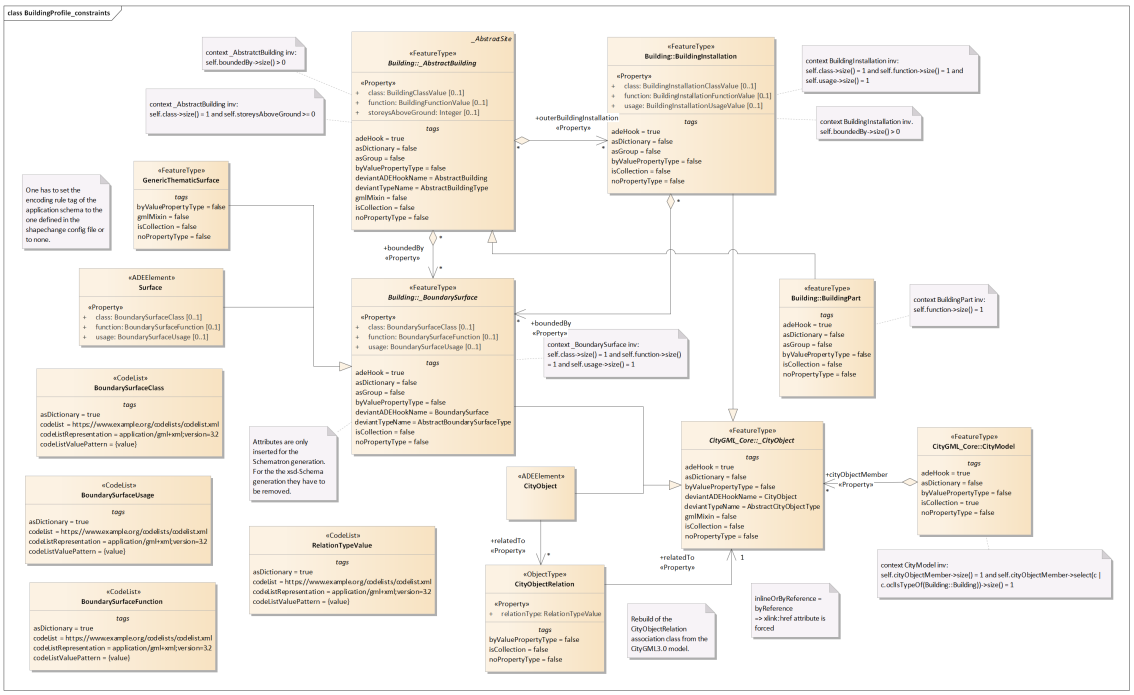


Abbildung 50 Klassendiagramm des Gebäude-Profils in CityGML 2.0.

Zusätzliche Bedingungen als OCL-Ausdrücke

Zusätzlich zur Modellierung der neuen Klassen bzw. neuen Attribute werden Bedingungen an das Gebäudemodell gestellt. Dazu werden den CityGML-Klassen OCL-Expressionen hinzugefügt (vgl. Abbildung 50). Diese definieren für die Unterklassen von *_AbstractBuilding Building, BuildingPart*, dass die Begrenzungsflächen als thematische Flächen bereitgestellt werden müssen. Außerdem sollen die Attribute *class* und *storeysAboveGround* im Datensatz enthalten sein.
Für die Elemente *BuildingInstallation* und die Unterklassen der thematischen Begrenzungsfläche *_BoundarySurface* gilt, dass die Attribute *class, function, usage* vorhanden sein sollen. Zusätzlich wird das Objekt *CityModel* auf ein Kind-Element beschränkt, welches vom Typ *Building* sein soll.
Die OCL-Ausdrücke werden im UML-Modell den Klassen hinzugefügt. Da diese OCL-Expressionen zum Großteil nicht durch ein XML-Schema realisiert werden können, werden die OCL-Bedingungen bei der XML-Schema-Erzeugung mittels *ShapeChange* in ein Schematron-Schema-Dokument übersetzt. Die resultierende Schematron-Schema-Datei ist in Listing A.2 in Anhang A zu sehen. Da die Attribute *class, function, usage* von *_BoundarySuface* als *_GenericApplicationPropertyOfBoundarySurface* modelliert werden, stehen diese im UML-Modell nicht für die OCL-Ausrücke zur Verfügung und müssen deshalb nach der Erzeugung des Schematron-Schemas manuell eingefügt werden.

Um die Attribute mit Code-Listenwert darauf zu überprüfen, ob sie einem Eintrag aus der Code-Liste, die im Attribut *codeSpace* referenziert wird, entspricht, müssen in der *ShapeChange*-Konfigurationsdatei die Regeln *rule-xsd-cls-codelist-constraints2, rule-xsd-cls-codelist-constraints-codeAbsenceInModelAllowed* zu den Enkodierungsregeln hinzugefügt werden. Außerdem müssen die *taggedValues codeList, codeListRepresentation, codeListValuePattern* der Codelist-Klassen gesetzt werden. XPath2.0 erlaubt es, externe XML-Dokumente durch XPath-Ausdrücke zu laden und das DOM (Document Object Model) des externen Dokuments weiter abzufragen.
So können die externen Code-Listen evaluiert werden, ob sie existieren, dem richtigen Format entsprechen, und der Wert des Attributs mit Code-Listen-Wert in dieser Code-Liste zu finden ist.
Zu beachten ist, dass bei CityGML 2.0 der GML-Namespace dem Schema von GML3.1 entspricht. Dieser Namensraum wird ebenfalls bei der Erzeugung des Schematron-Schemas verwendet. Wird anschließend ein CityGML 2.0-Dokument, welches Code-Listen im GML3.2-Format referenziert, mittels der Schematron-Datei validiert, kommt es zu Fehlermeldungen, dass die referenzierte Code-Liste nicht im richtigen Format vorliegt. Das liegt daran, dass die GML-Namensräume des CityGML 2.0-Dokuments und des Code-Listen-Dokuments nicht übereinstimmen. Es müssen also für CityGML 2.0-Datensätze Code-Listen-Dokumente in der selben GML-Version verwendet werden. Um bzgl. der Code-Listen unabhängig von der GML-Version zu werden ist es möglich, die entsprechenden Bereiche der Schematron-Datei auszukommentieren.

Validierung mittels Schematron

Auf der offiziellen Schematron-Website (`https://schematron.com/front-page/the-schematron-skeleton-implementation/` bzw. `https://github.com/Schematron/schematron/tree/master/trunk/schematron/code`) ist die Vorgehensweise von der Konvertierung eines Schematron-Schemas in ein XSLT-Stylesheet bis zur Validierung eines Dokuments beschrieben:

1) Vorverarbeitung des Schematron-Schemas mit der Datei *iso_dsdl_include.xsl*, um das Schema aus verschiedenen Schema-Teilen zu vereinen. Wenn das Schema nicht in separaten Teilen vorliegt, kann dieser Schritt übersprungen werden.

2) Das (Ergebnis-)Dokument aus dem ersten Schritt wird mit dem Stylesheet *iso_abstract_expand.xsl* transformiert. Dabei werden abstrakte *Patterns* in reale *Patterns* konvertiert. Enthält das Eingangsdokument keine abstrakte *Patterns*, so kann dieser Schritt ausgelassen werden.

3) Kompilierung bzw. Transformation des Schematron-Schemas in ein XSLT-Skript mittels einer der zwei Dateien *iso_svrl_for_xslt1.xsl, iso_svrl_for_xslt2.xsl*.

4) Validierung des XML-Dokuments mit dem XSLT-Stylesheet aus Schritt 3). Das Ausgangsdokument liegt im SVRL (Schematron Validation Report Language)-Format vor und kann nun in weitere Datenformate transformiert werden, z.B. in das HTML-Format für eine Ansicht im Browser.

(`https://github.com/Schematron/schematron/tree/master/trunk/schematron/code`)

Über die offiziellen XSL-Dateien kann das Schematron-Schema-Dokument aus Listing A.2 mittels XSL-Transformationen in ein XSLT-Dokument umgewandelt werden, das anschließend auf das zu validierende XML-Dokument angewandt wird.
Als XSLT2.0/XPath2.0-fähiger XSLT-Prozessor steht bspw. der *Saxon-HE*-Prozessor zur Verfügung (`https://www.saxonica.com/welcome/welcome.xml`). Dieser kann aus dem entsprechenden Maven-Repository heruntergeladen werden (`https://mvnrepository.com/artifact/net.sf.saxon/Saxon-HE`).

In Listing 4.1 ist ein beispielhaftes *batch*-Skript zur Validierung eines Datensatzes gezeigt. Zunächst wird das Schematron-Schema mittels des offiziellen Stysheet *iso_svrl_for_xslt2.xsl* in eine XSL-Skript-Datei umgewandelt. Das XSL-Skript für die Validierung von CityGML 2.0-Dateien ist in Listing A.3 in Anhang A einsehbar.
Anschließend wird diese XSL-Datei auf das zu überprüfende XML-Dokument angewandt, um einen Bericht im SVRL-Format zu erzeugen. Der resultierende Fehler-Bericht kann über ein zusätzliches XSLT-Stylesheet (siehe Listing 4.2) in eine HTML-Datei für die Ansicht im Browser transformiert werden.

```
java -jar Saxon-HE-9.9.1-6.jar -t -s:path/to/schematronFile.sch -xsl:path/to/iso_svrl_for_xslt2.xsl -o:outputpath/for/
    validationScript.xsl
java -jar Saxon-HE-9.9.1-6.jar -t -s:path/to/xmlDocument.xml -xsl:path/to/validationScript.xsl -o:outputpath/for/report.xml
java -jar Saxon-HE-9.9.1-6.jar -t -s:path/to/report.xml -xsl:path/to/SVRLReportRender.xsl -o:outputpath/for/report.html
```

Listing 4.1: xslt.bat

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:svrl="http://purl.oclc.org/dsdl/svrl">
  <xsl:output method="html" indent="yes"/>

  <xsl:template match="/">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>SVRL report</title>
        <style>
          html {
            font-family: sans-serif;
          }

          b span {}

          td {
            border: 2px solid black;
            text-align: left;
            padding: 8px;
          }

          td.inner {
            border: 2px solid #dddddd;
            text-align: left;
          }
        </style>

        <script type='text/javascript' src="../Saxon-JS-2.0/SaxonJS2.js">
        </script>

        <script type="text/javascript" src="../js/EventFunctions.js">
        </script>

      </head>
      <body>
        <h1>SVRL report</h1>
        <div>Validated file: <span id="filepath"> <xsl:value-of select=".//svrl:active-pattern/@document"/> </span> </div>
        <div>Errors: <xsl:value-of select="count(.//svrl:failed-assert)"/> </div>
        <!-- <xsl:apply-templates select=".//svrl:failed-assert"/> -->
        <table>
        <xsl:for-each select=".//svrl:failed-assert">
          <tr>
            <td>
              <table>
                <tr>
                  <td class="inner">
                    <b style="color: red;">Error: </b>
                    <span style="color: red;"> <xsl:value-of select="svrl:text"/> </span>
                  </td>
                </tr>
                <tr>
                  <td class="inner">
                    <b>Test: </b> <xsl:value-of select="@test"/>
                  </td>
                </tr>
                <tr>
                  <td class="inner xpath-location"> <b>(XPath-)Location in the document: </b> <span class="
      xpath-location-text"> <xsl:value-of select="@location"/> </span> </td>
                </tr>
                <tr style="border: none; padding: 0;">
                  <td style="border: none; padding: 0;"> <button onclick="showElementFunction(event);" class="
      showElementButton" type="button" disabled="true">Show element</button> </td>
                </tr>
              </table>
            </td>
          </tr>
        </xsl:for-each>
        </table>
        <p id="date"></p>
        <script type="text/javascript" src="../js/Setup.js">
        </script>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="svrl:failed-assert">
    <div class="result-assert">
      <div class="result-assert-test">
        <span class="label">
          <b>Test: </b>
```

```
        </span>
        <xsl:value-of select="@test" />
      </div>
      <div class="result-assert-location">
        <span class="label">
          <b>Location: </b>
        </span>
        <xsl:value-of select="@location" />
      </div>
      <div class="result-assert-text">
        <span class="label">
          <b>Description: </b>
        </span>
        <xsl:value-of select="svrl:text" />
      </div>
    </div>
  </xsl:template>
</xsl:stylesheet>
```

Listing 4.2: svrl2html.xsl

Im Folgenden wird *validationScript.xsl* auf einen einfachen Beispieldatensatz *example.gml* angewandt und anschließend eine HTML-Datei des Fehler-Berichts erzeugt. Der Datensatz enspricht der geforderten Modellierung des Gebäudes mit thematischen Begrenzungsflächen. Allerdings werden als Test verschiedene Fehler in den Datensatz integriert, die in Listing 4.3 mittels Kommentare dokumentiert sind. Diese Fehler sollen durch die Validierung aufgedeckt werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Original-Datei: Beispiel SimpleBuilding-lod3-SrefBS.xml Building mit Solid-Geometrie und BoundarySurfaces als
     MultiSurface-Geometrie Autor: A. Koukofikis, V. Coors -->
<core:CityModel  xmlns="http://www.opengis.net/citygml/profiles/base/2.0"
        xmlns:core="http://www.opengis.net/citygml/2.0"
        xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
        xmlns:gml="http://www.opengis.net/gml"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:buildingProfile="http://www.citygml.org/buildingProfile"
        xsi:schemaLocation="http://www.citygml.org/buildingProfile file:///C:/Users/felix/Documents/ShapeChange-2.9.0/
     BuildingProfile_CityGML2.0/BuildingProfile_CityGML2.0/xsds/INPUT/BuildingProfile.xsd">
  <!-- Second building in the city model -->
  <core:cityObjectMember>
    <bldg:Building/>
  </core:cityObjectMember>
  <core:cityObjectMember>
      <bldg:Building gml:id="SimpleLod3Building">
      <!-- Empty class attribute -->
      <bldg:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml"></bldg:class>
      <bldg:storeysAboveGround>3</bldg:storeysAboveGround>
      <bldg:boundedBy>
        <bldg:WallSurface gml:id="SimpleLod3Building_ws-2">
          <bldg:lod3MultiSurface>
            <gml:MultiSurface srsDimension="3">
              ...
            </gml:MultiSurface>
          </bldg:lod3MultiSurface>
          <!-- Invalid code list -->
          <buildingProfile:class codeSpace="https://www.sig3d.de/codelists/Handbuch-SIG3D/building/2.0/CL-V1.0/
     _AbstractBuilding_class.xml">2000</buildingProfile:class>
          <!-- Not existing code list value -->
          <buildingProfile:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.
     xml">notInCodeList</buildingProfile:function>
          <!-- Not existing code list -->
          <buildingProfile:usage codeSpace="https://www.sig3d.de/notExistingCodeList.xml">2000</buildingProfile:usage>
        </bldg:WallSurface>
      </bldg:boundedBy>
      <bldg:boundedBy>
        <bldg:RoofSurface gml:id="SimpleLod3Building_rs-1">
          <bldg:lod3MultiSurface>
            <gml:MultiSurface srsDimension="3">
              ...
            </gml:MultiSurface>
          </bldg:lod3MultiSurface>
        </bldg:RoofSurface>
```

```
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:WallSurface gml:id="SimpleLod3Building_ws-3">
        <bldg:lod3MultiSurface>
          <gml:MultiSurface srsDimension="3">
            ...
          </gml:MultiSurface>
        </bldg:lod3MultiSurface>
        <!-- Duplicate attribute -->
        <buildingProfile:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry0</buildingProfile:class>
        <buildingProfile:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry0</buildingProfile:class>
        <!-- Not existing code list value -->
        <buildingProfile:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">notExistingCodeListValue</buildingProfile:function>
        <buildingProfile:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry0</buildingProfile:usage>
      </bldg:WallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:WallSurface gml:id="SimpleLod3Building_ws-1">
        <bldg:lod3MultiSurface>
          <gml:MultiSurface srsDimension="3">
            ...
              </gml:MultiSurface>
            </bldg:lod3MultiSurface>
          </bldg:Window>
        </bldg:opening>
        <bldg:opening>
          <bldg:Door gml:id="SimpleLod3Building_ws-1_d-1-1">
            <bldg:lod3MultiSurface>
              <gml:MultiSurface srsDimension="3">
                ...
              </gml:MultiSurface>
            </bldg:lod3MultiSurface>
          </bldg:Door>
        </bldg:opening>
        <buildingProfile:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry0</buildingProfile:class>
        <buildingProfile:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry1</buildingProfile:function>
        <!-- Missing 'usage' element -->
      </bldg:WallSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:GroundSurface gml:id="SimpleLod3Building_gs-1">
        <bldg:lod3MultiSurface>
          <gml:MultiSurface srsDimension="3">
            ...
          </gml:MultiSurface>
        </bldg:lod3MultiSurface>
      </bldg:GroundSurface>
    </bldg:boundedBy>
    <bldg:boundedBy>
      <bldg:WallSurface gml:id="SimpleLod3Building_ws-4">
        <bldg:lod3MultiSurface>
          <gml:MultiSurface srsDimension="3">
            ...
          </gml:MultiSurface>
        </bldg:lod3MultiSurface>
        <!-- Everything valid -->
        <buildingProfile:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry0</buildingProfile:class>
        <buildingProfile:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry1</buildingProfile:function>
        <buildingProfile:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml">codeListEntry1</buildingProfile:usage>
      </bldg:WallSurface>
    </bldg:boundedBy>
    <bldg:lod3Solid>
      <gml:Solid>
        <gml:exterior>
          <gml:CompositeSurface>
            <gml:surfaceMember xlink:href="#UUID_91a3c2f0-4ead-4dc7-91c5-8159479fe378" />
            <gml:surfaceMember xlink:href="#UUID_b57973dc-bf29-4708-8e60-68dac0bdc533" />
            <gml:surfaceMember xlink:href="#UUID_976313a5-d9ca-4e59-b6cb-cf7bf3754184" />
            <gml:surfaceMember xlink:href="#UUID_f695710b-2d22-48b4-8798-cc8bdbcbf521" />
            <gml:surfaceMember xlink:href="#UUID_249d0771-a719-40ed-b133-c070f8539d8f" />
            <gml:surfaceMember xlink:href="#UUID_34cd562b-acfd-41b5-a92a-6af252b39ad9" />
            <gml:surfaceMember xlink:href="#UUID_008101e8-3546-488a-95e0-9ef98b7d2065" />
            <gml:surfaceMember xlink:href="#UUID_4c21bfb9-5704-4d92-aa3b-5dc397053353" />
```

```
116                 <gml:surfaceMember xlink:href="#UUID_93a10d9e-1896-41a2-b025-5396f6b6dce5" />
117                 <gml:surfaceMember xlink:href="#UUID_e1073b6a-6fae-4f8a-ae2e-6812cad4ca20" />
118                 <gml:surfaceMember xlink:href="#UUID_a07b2848-8ec5-487f-84ad-500ddc63b51b" />
119                 <gml:surfaceMember xlink:href="#UUID_3e1c8e9a-4c15-4a32-bde8-a55b20fa4554" />
120                 <gml:surfaceMember xlink:href="#UUID_40bbbc27-1cfa-4c02-a892-496bdf2744d5" />
121                 <gml:surfaceMember xlink:href="#UUID_158fde4d-11f9-4448-8eb4-be2129dc0ac6" />
122                 <gml:surfaceMember xlink:href="#UUID_f3bd6cc5-2aec-44b5-8b3a-99a9fb6d6be9" />
123                 <gml:surfaceMember xlink:href="#UUID_fb11d028-3ee6-4b39-b1cc-e822529d6ffe" />
124               </gml:CompositeSurface>
125             </gml:exterior>
126           </gml:Solid>
127         </bldg:lod3Solid>
128       </bldg:Building>
129     </core:cityObjectMember>
130 </core:CityModel>
```

Listing 4.3: example.gml

Wird die XSLT-Transformation mit dem Stylesheet *validationScript.xsl* auf das Dokument *example.gml* angewandt, wird der SVRL-Bericht *validationReport.xml* (siehe Anhang A) erhalten.

Der Fehlerbericht wird nun über *svrl2html.xsl* für eine ansprechende Visualisierung der Fehlermeldung und Anzeige der fehlerhaften Elemente in eine HTML-Datei transformiert. Dort werden die Fehler tabellarisch aufgelistet. Zudem ist es möglich, sich die fehlerhaften XML-Elemente anzeigen zu lassen, um diese zu inspizieren. In Abbildung 51) ist beispielhaft dargestellt, wie das *<bldg:WallSurface>*-Element begutachtet wird, welches eine unzulässige Anzahl an *<buildingProfile:class>*-Elementen beinhaltet.

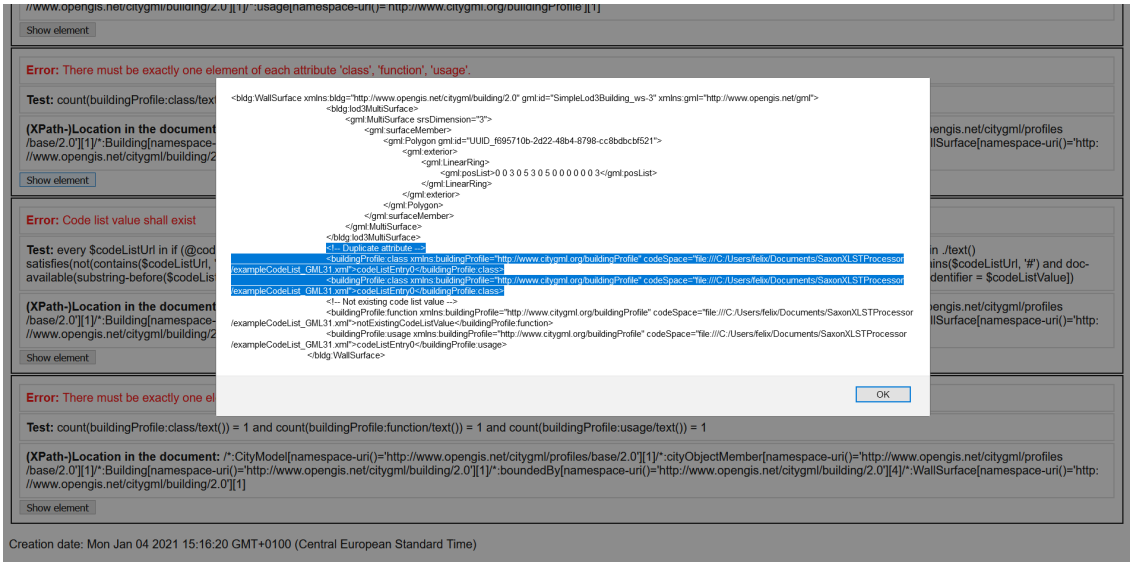


Abbildung 51 Anzeige des fehlerhaften Elements im HTML-Validierungsbericht.

Gebäude-Profil für CityGML 3.0

UML-Modellierung

Datensätze von Gebäuden im Format der CityGML-Version 3.0 sollten nach dem Gebäudeprofil für die Eingangsdaten (siehe Abbildung 52) modelliert werden. Gebäudemodelle im CityGML 3.0-Format weisen damit dieselben Attribute wie Datensätze im CityGML 2.0-Format auf. Dies macht die Implementierung eines Algorithmus möglich, der Datensätze beider Versionen des CityGML-Standards verarbeiten kann.
Die zusätzlichen Attribute zur Charakterisierung einer thematischen Gebäudefläche wird über die Klasse *SurfaceProperties* realisiert. Das resultierende Gebäudeprofil *BuildingProfile.xsd* importiert die Schema-Dateien *building.xsd*, sowie die Schema-Datei *BuildingADE.xsd* der Erweiterung des *Building*-Moduls (siehe Anhang B). Im Instanz-Dokument wird dann alleinig das *BuildingProfile.xsd* importiert. Eine Alternative ist die *Inline*-Definition des Gebäudeprofils im Instanz-Dokument (siehe *building.gml*).

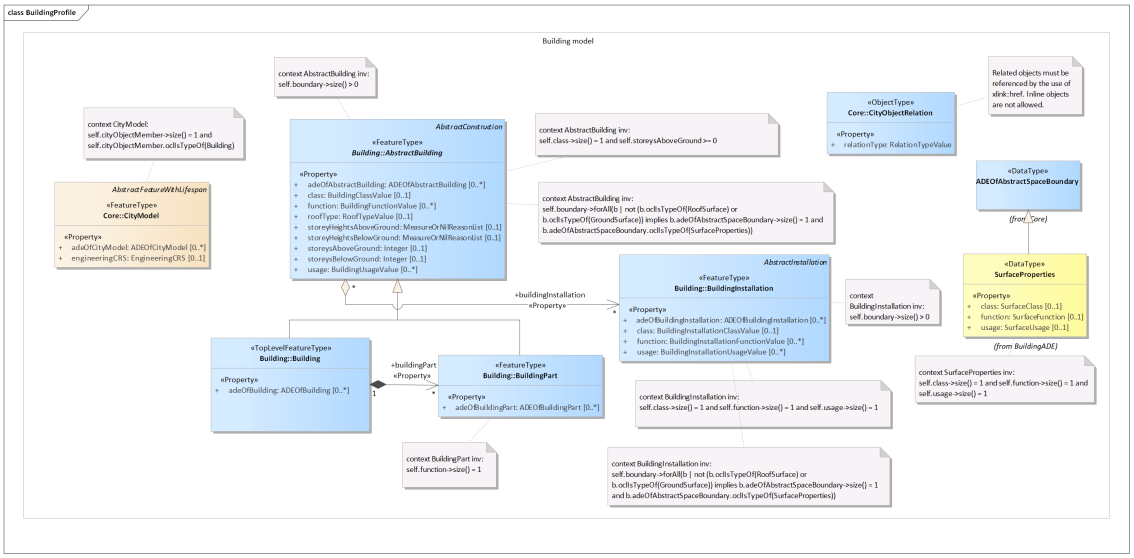


Abbildung 52 Klassendiagramm des CityGML 3.0-Gebäudemodells mit OCL-Bedingungen.

Zusätzliche Bedingungen als OCL-Ausdrücke

Die Kompatibilität zwischen den Gebäudemodellen der CityGML-Versionen 2.0 und 3.0 impliziert, dass das CityGML 3.0-Gebäude-Profil dieselben Bedingungen an das Gebäudemodell stellt, wie das für CityGML 2.0-Gebäudemodelle der Fall ist. Die OCL-Ausdrücke unterscheiden sich aufgrund der Unterschiede der Datenmodelle in den Formulierungen, stellen jedoch den gleichen semantischen Inhalt des Gebäudemodells sicher. Das resultierende Schematron-Schema ist in Listing B.3 in Anhang B zu sehen. Auch hier werden die referenzierten Code-Listen und Code-Listen-Werte auf Existenz überprüft. Wie schon bei dem Gebäudeprofil für CityGML 2.0 muss darauf geachtet werden, dass die referenzierten Code-Listen im richtigen GML-Format vorliegen, um keine Fehlermeldungen zu bekommen. Da CityGML 3.0 auf GML3.2 basiert, müssen dies auch die verwendeten Code-Listen. Um bzgl. der Code-Listen unabhängig von der GML-Version zu werden ist es möglich, die entsprechenden Bereiche der Schematron-Datei auszukommentieren.

Nach der in Abschnitt 4.2.2 beschriebene Anwendung von Transformationsschritten liegt die Datei *validationScriptCityGML3.0.xsl* (siehe Anhang B) für die Validierung des Beispieldatensatzes *building.gml* vor.

Auch in diesem Datensatz sind Fehler enthalten (siehe Kommentare in *building.gml*), um die Funktionsweise der Schematron-Validierung zu testen. Nach der Validierung liegen der SVRL-Report *validationReportCityGML3.0.xml* (siehe Anhang B), sowie der Fehlerbericht für die Browser-Ansicht vor.

```
<?xml version="1.0" encoding="UTF-8"?>
<core:CityModel xmlns:clrnceSpce="http://www.citygml.org/profile/building/clearanceSpaceComputation" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:core="http://www.opengis.net/citygml/3.0" xmlns:con="http://www.opengis.net/citygml/construction/3.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:bldg="http://www.opengis.net/citygml/building/3.0" xsi:schemaLocation="http://www.citygml.org/profile/building/clearanceSpaceComputation ../xsdFiles/citygml3.0/BuildingADE.xsd http://www.opengis.net/citygml/3.0 ../xsdFiles/citygml3.0/cityGMLBase.xsd http://www.opengis.net/citygml/building/3.0 ../xsdFiles/citygml3.0/building.xsd http://www.opengis.net/citygml/construction/3.0 ../xsdFiles/citygml3.0/construction.xsd">
  <!-- Too many child elements in CityModel element -->
  <core:versionMember/>
  <core:cityObjectMember>
    <bldg:Building gml:id="GMLID_BUI303778_1460_14976">
      <core:boundary>
        <core:ClosureSurface gml:id="UUID_8c75aba4-e1a4-432f-ad76-417706c323a1">
          <core:adeOfAbstractSpaceBoundary>
            <clrnceSpce:SurfaceProperties>
              <clrnceSpce:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:class>
              <clrnceSpce:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:function>
              <clrnceSpce:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:usage>
            </clrnceSpce:SurfaceProperties>
          </core:adeOfAbstractSpaceBoundary>
          <!-- Too many 'adeOfAbstractSpaceBoundary' elements -->
          <core:adeOfAbstractSpaceBoundary>
            <clrnceSpce:SurfaceProperties>
              <clrnceSpce:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:class>
              <clrnceSpce:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:function>
              <clrnceSpce:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:usage>
            </clrnceSpce:SurfaceProperties>
          </core:adeOfAbstractSpaceBoundary>
          <core:lod3MultiSurface>
            ...
          </core:lod3MultiSurface>
        </core:ClosureSurface>
      </core:boundary>
      <core:boundary>
        <con:GroundSurface gml:id="fme-gen-b799466a-f967-42ca-be44-236e7f51bc18">
          <core:lod3MultiSurface>
            ...
          </core:lod3MultiSurface>
        </con:GroundSurface>
      </core:boundary>
      <core:boundary>
        <con:RoofSurface gml:id="UUID_85ddd1e8-fb8f-433f-b39a-7f97a224ac3d">
          <core:lod3MultiSurface>
            ...
          </core:lod3MultiSurface>
        </con:RoofSurface>
      </core:boundary>
      <core:boundary>
        <con:WallSurface gml:id="UUID_3de79848-2577-487e-96f0-49775008abd0">
          <core:adeOfAbstractSpaceBoundary>
            <clrnceSpce:SurfaceProperties>
              <clrnceSpce:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:class>
              <clrnceSpce:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:function>
              <clrnceSpce:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:usage>
            </clrnceSpce:SurfaceProperties>
          </core:adeOfAbstractSpaceBoundary>
          <core:lod3MultiSurface>
```

```
            ...
          </core:lod3MultiSurface>
        </con:WallSurface>
      </core:boundary>
      <core:boundary>
        <con:WallSurface gml:id="UUID_4d55f350-0bb7-46e7-90e9-2f81ee2390df">
          <core:adeOfAbstractSpaceBoundary>
            <clrnceSpce:SurfaceProperties>
              <clrnceSpce:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:class>
              <clrnceSpce:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:function>
              <clrnceSpce:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</clrnceSpce:usage>
            </clrnceSpce:SurfaceProperties>
          </core:adeOfAbstractSpaceBoundary>
          <core:lod3MultiSurface>
            ...
          </core:lod3MultiSurface>
        </con:WallSurface>
      </core:boundary>
      ...
      <!-- Missing 'storeysAboveGround' attribute -->
      <bldg:class codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/buildingClass.gml">1000</bldg:class>
      ...
      <bldg:buildingInstallation>
        <bldg:BuildingInstallation gml:id="UUID_931a2987-e042-4cd9-a61c-8f446bcbdf96">
          <core:relatedTo>
            <core:CityObjectRelation>
              <core:relationType codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/relationTypeValue.gml">2000</core:relationType>
              <!-- Invalid inline element of related object -->
              <core:relatedTo>
                <con:WallSurface/>
              </core:relatedTo>
            </core:CityObjectRelation>
          </core:relatedTo>
          <core:boundary>
            <con:RoofSurface gml:id="UUID_bf8d40bc-e91e-43bd-93d7-3de662e9fff6">
              <core:lod3MultiSurface>
                ...
              </core:lod3MultiSurface>
            </con:RoofSurface>
          </core:boundary>
          <bldg:class codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/buildingInstallationClass.gml">codeListEntry0</bldg:class>
          <bldg:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</bldg:function>
          <bldg:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</bldg:usage>
        </bldg:BuildingInstallation>
      </bldg:buildingInstallation>
      ...
      <bldg:buildingInstallation>
        <bldg:BuildingInstallation gml:id="UUID_372c77ac-c0f5-4aaf-9e74-90bf860f99c2">
          <core:relatedTo>
            <core:CityObjectRelation>
              <core:relationType codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/relationTypeValue.gml">2000</core:relationType>
              <core:relatedTo xlink:href="#UUID_3de79848-2577-487e-96f0-49775008abd0"/>
            </core:CityObjectRelation>
          </core:relatedTo>
          <core:boundary>
            <con:RoofSurface gml:id="UUID_14df853d-7c0d-48b3-b87f-c3991c125df3">
              <core:lod3MultiSurface>
                ...
              </core:lod3MultiSurface>
            </con:RoofSurface>
          </core:boundary>
          <!-- Invalid code list value -->
          <bldg:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">2000</bldg:class>
          <bldg:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</bldg:function>
          <bldg:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">codeListEntry0</bldg:usage>
        </bldg:BuildingInstallation>
      </bldg:buildingInstallation>
      <bldg:buildingInstallation>
        <bldg:BuildingInstallation gml:id="UUID_c22c04a2-678d-4200-ba2d-8ea114b7e7fd">
          <core:relatedTo>
            <core:CityObjectRelation>
```

```
124             <core:relationType codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/relationTypeValue.gml">5000</
    core:relationType>
125             <core:relatedTo xlink:href="#UUID_6efc2b4c-4190-4cec-ad34-88dd42e487eb"/>
126           </core:CityObjectRelation>
127         </core:relatedTo>
128         <core:boundary>
129           <core:ClosureSurface gml:id="UUID_c5c61a86-342c-4a93-94b7-6829f41592d0">
130             <core:adeOfAbstractSpaceBoundary>
131               <clrnceSpce:SurfaceProperties>
132                 <clrnceSpce:class codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
    xml">codeListEntry0</clrnceSpce:class>
133                 <clrnceSpce:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32
    .xml">codeListEntry0</clrnceSpce:function>
134                 <clrnceSpce:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
    xml">codeListEntry0</clrnceSpce:usage>
135               </clrnceSpce:SurfaceProperties>
136             </core:adeOfAbstractSpaceBoundary>
137             <core:lod3MultiSurface>
138               ...
139             </core:lod3MultiSurface>
140             <!-- Missing 'class', 'function', 'usage' attribute -->
141           </core:ClosureSurface>
142         </core:boundary>
143         ...
144         <!-- Invalid GML3.1 code list -->
145         <bldg:class codeSpace="file:///C:/Users/felix/Documents/CityGML_Codelists/buildingInstallationClass.gml">1000</
    bldg:class>
146         <bldg:function codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">
    codeListEntry0</bldg:function>
147         <bldg:usage codeSpace="file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml">
    codeListEntry0</bldg:usage>
148       </bldg:BuildingInstallation>
149     </bldg:buildingInstallation>
150   </bldg:Building>
151 </core:cityObjectMember>
152 </core:CityModel>
```

Listing 4.4: building.gml

Der resultierende Fehler-Bericht kann im Browser betrachtet werden. Abbildung 53 zeigt bspw. die Inspektion eines fehlerhaften zweiten *Child*-Elements des Wurzel-Elements *CityModel*.

Abbildung 53 Inspektion des fehlerhaften Kind-Element des *Root*-Elements *CityModel* im HTML-Report.

Verwendung von Code-Listen

Der implementierte Algorithmus basiert auf der Auswertung der zusätzlichen Attribute, welche über die vorher beschriebenen ADEs und Profile zur Verfügung stehen. Die jeweiligen Code-Listen-Attribute müssen, falls sie in Objekten vorhanden sind, für die Funktionsweise des Algorithmus bestimmte Werte annehmen. Im Folgenden sind die Code-Listen aufgeführt, die für die eingehenden Gebäudemodelle obligatorisch sind.

Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” thematischer Gebäudeflächen

class	***function***	***usage***
facade	gable	relevantForClearanceSpace
sidewall		privileged

Tabelle 9 Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” thematischer Gebäudeflächen.

- CityGML 2.0-Profil: *class-, function-, usage*-Attribute werden über generische Attribut-Elemente mit dem Präfix *buildingProfile* bereitgestellt
- CityGML 3.0-Profil: *class-, function-, usage*-Attribute werden innerhalb der ADE-Klasse *SurfaceProperties* bereitgestellt

Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” für Gebäudeinstallationen

class	***function***	***usage***
primaryElement	structuralElement	roofOverhang
secondaryElement	porch	cornice
	roofConstruction	chimney
		balcony
		dormer
		loggia

Tabelle 10 Zulässige Code-Listen-Werte der Attribute ”class”, ”function”, ”usage” für Gebäudeinstallationen.

- CityGML 2.0-Profil: *class-, function-, usage*-Attribute werden über die standardmäßigen Attribut-Elemente der Klasse *BuildingInstallation* des CityGML-Standards bereitgestellt
- CityGML 3.0-Profil: *class-, function-, usage*-Attribute werden über die standardmäßigen Attribut-Elemente der Klasse *BuildingInstallation* des CityGML-Standards bereitgestellt

Code-Listen-Werte der Attribute ”class”, ”function” für Gebäude/Gebäudeteile

class	***function***
buildingClass1	garage
buildingClass2	
buildingClass3	

Tabelle 11 Zulässige Code-Listen-Werte der Attribute ”class”, ”function” für Gebäude/Gebäudeteile.

- CityGML 2.0-Profil: *class-, function*-Attribute werden über die standardmäßigen Attribut-Elemente der Klasse *Building* des CityGML-Standards bereitgestellt
- CityGML 3.0-Profil: *class-, function*-Attribute werden über die standardmäßigen Attribut-Elemente der Klasse *Building* des CityGML-Standards bereitgestellt

Zulässige Code-Listen-Werte des Attributs ”relationType” für Objekt-Relationen

relationType
adjacentRoofAbove
referenceComponentForProtrusion
referenceComponentForRoofConstruction
adjacentRoofConstruction
subjacentRoof
adjacentBuildingInstallation

Tabelle 12 Code-Listen-Werte des Attributs ”relationType” für Objekt-Relationen.

- CityGML 2.0-Profil: *relationType*-Attribut wird über das Attribut-Element der Klasse *buildingProfile::CiyObjectRelation* der ADE bereitgestellt
- CityGML 3.0-Profil: *relationType*-Attribut wird über das standardmäßigen Attribut-Element der Klasse *CityObjectRelation* des CityGML-Standards bereitgestellt

Prinzipieller Ablauf der Datensatz-Validierung

Prinzipiell besteht der Validierungsprozess eines Datensatzes aus zwei Phasen. In Phase 1 wird das XML-Dokument gegen die XSD-Schema-Dateien geprüft, auf welchen das Instanz-Dokument basiert. Im Falle eines Gebäudemodells, dass für die Abstandsflächenberechnung verwendet werden soll, wäre es das jeweilige CityGML-Profil, welches die nötigen Schema-Dateien importiert. Würde bspw. in einem *<WallSurface>*-Element ein *<adeOfAbstractThematicSurface>*-Element enthalten sein, so würde die Validierung einen Fehler feststellen, da laut CityGML 3.0-Schema der Typ *ADEOfAbstractThematicSurface* abstrakt ist von einer ADE mittels einer Unterklasse instanziierbar gemacht werden muss. Da die Definition einer Unterklasse von *ADEOfAbstractThematicSurface* jedoch in den Schema-Dateien des Gebäudeprofils nicht enthalten ist, würde das Dokument nicht validieren.
Ist laut XML-Schema ein valides Dokument erreicht, wird in Phase 2 getestet, ob die OCL-Konditionen von dem Dokument eingehalten werden. Wird auch das erreicht, wird das Dokument als gültig betrachtet.

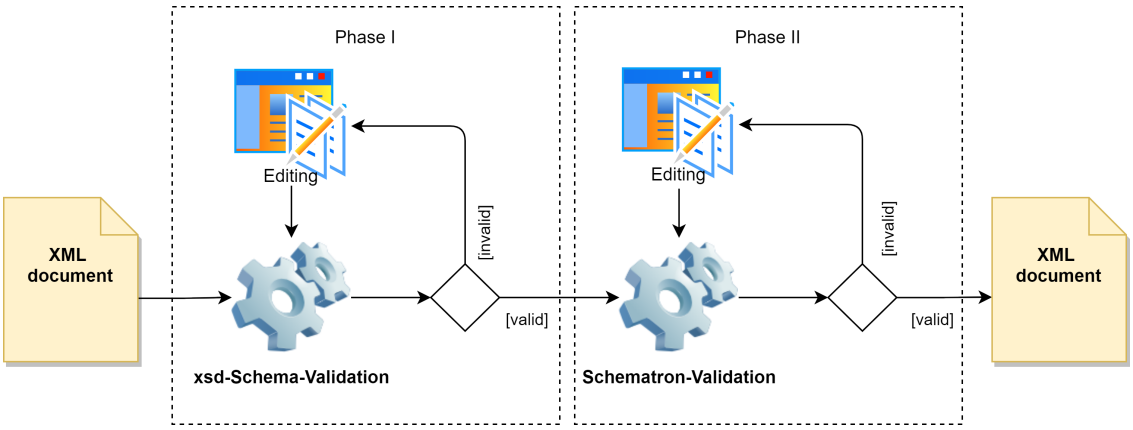


Abbildung 54 Zwei-phasiger Validierungsprozess eines XML-Dokuments mit XML-Schema- und Schematron-Validierung.

Modellierungsparadigmen bzgl. des Abstandsflächenalgorithmus

Damit der in dieser Arbeit implementierte Algorithmus korrekte Abstandsflächen berechnen kann, sind einige Dinge bei der Modellierung eines CityGML-Gebäudes zu beachten:

- Alle für die Abstandsflächen relevant erachteten Bauteile, die nicht eindeutig als Außenwand oder Dach identifiziert werden können, sollen als Gebäudeinstallation modelliert und mittels der Code-Listen-Attribute (vgl. Abschnitt 4.2.2) näher spezifiziert werden. Dazu gehören dekorative Bauelemente wie bspw. Gesimse, Dachüberstände, o.ä..

Abbildung 55 Gesimse sollen nicht als Teil der Außenwand, sondern als dekorative Bauelemente, also als Gebäudeinstallationen, modelliert werden (Poellet, 2008).

- Gebäudeinstallationen, die sich über mehrere Referenzobjekte erstrecken (bspw. Balkone, die sich über zwei oder mehr Gebäudeseiten erstrecken), sollten in mehrere individuelle Gebäudeinstallationen unterteilt werden, sodass eine 1:1-Abbildung der Gebäudeinstallation auf ein Referenzobjekt entsteht.

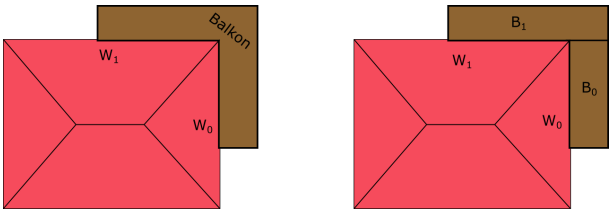


Abbildung 56 Draufsicht auf ein Gebäude mit Balkon. Ein Balkon der sich über zwei Gebäudeseiten W_0 und W_1 erstreckt (links), soll in zwei individuelle Balkone unterteilt werden, um diese mittels einer 1:1-Beziehung mit den Gebäudeseiten referenzieren zu können (rechts).

- Von der Modellierung von Gebäudeinstallationen in CityGML 2.0 durch reine Geometrie wird abgeraten und würde durch die in der Schematron-Validierung geforderte Modellierung kein valides Dokument darstellen. Für nicht klassifizierbare Flächen soll die ADE-Klasse *GenericThematicSurface* verwendet werden.

- Gebäudeteile (*BuildingPart*) sollten als *Inline*-Definition im Gebäude selbst definiert werden, da die Schematron-Validierung nur ein einziges Kind-Element des Elements *CityModel* erlaubt.

- Die dominierenden Referenzobjekte der Gebäudeinstallationen sollen über die Klasse *CityObjectRelation* referenziert werden und über das Attribut ”relationType” die Art der Relation spezifizieren (vgl. Abschnitt 4.2.2). Zwar implementiert der Algorithmus eine topologische Analyse, es kann jedoch nicht ausgeschlossen werden, dass unter Umständen nicht die gewünschten Komponenten als Referenzobjekte identifiziert werden, wenn diese rein geometrisch nicht eindeutig zuordbar sind, sondern eine Gesamtimpression des Gebäudekontexts erfordern. Dies ist v.a. der Fall, wenn Gebäudeinstallationen an mehrere übergeordnete Elemente angrenzen.

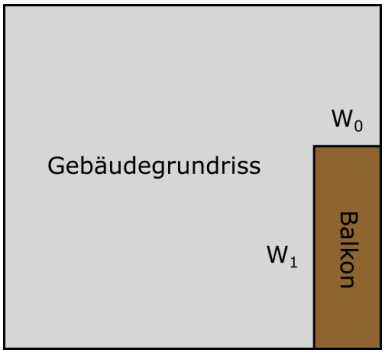


Abbildung 57 Der Balkon grenzt an die Beiden Außenwände W_0 und W_1 an. Das dominierende Referenzobjekt soll über die Klasse *CityObjectRelation* definert werden.

- Dachflächen werfen nach BauO keine eigenen Abstandsflächen. Das Attribut ”usage” von Dachflächen sollte deshalb wenn möglich nicht vorhanden sein, oder wenn doch den Wert ”privileged” aufweisen (vgl. Abschnitt 4.2.2).

4.2.3 Programm-internes Datenmodell

CityGML-Building-Modul

Um das CityGML-Gebäudemodell zur Laufzeit im Programm in interner Repräsentation darstellen zu können, wird das *Building*-Modul des CityGML-Standards und dessen thematische Flächenobjekte in vereinfachter und modifizierter Form nachempfunden (siehe Abbildung 58 und Abbildung 59). Dabei wird versucht, Kompatibilität sowohl zu CityGML Version 2.0 als auch zu Version 3.0 zu schaffen.
Für eine Dokumentation im Detail wird auf Anhang Anhang D verwiesen.

Thematische Gebäudeteilflächen

Im Gegensatz zum CityGML-Standard wird im internen Datenmodell für die Abstandsflächenberechnung jede Flächengeometrie einer Multi-Flächgeometrie als eigenständiges thematisches Teilflächenobjekt modelliert. So besteht bspw. ein *WallSurface*-Objekt nicht aus einer Multi-Flächengeometrie mit rein geometrischen Teilflächen, sondern aus thematischen Teilflächenelementen. Das erlaubt es, für jedes Teilflächenobjekt neben der Geometrie zusätzliche Attribute bereitzustellen. So können bspw. über das Attribut ”roofEdges” die adjazenten Kanten einer Dachfläche mit Referenz zum jeweiligen Dachflächenobjekt bereitgestellt werden.
Da jedes Flächenteil in eigenen Abstandsflächen resultieren kann (jeder einzelne Wandteil resultiert in einer Abstandsfläche [vgl. Abschnitt 4.1.1]), besteht eine Assoziation zwischen jeder der thematischen Teilflächenklassen und der Abstandsflächenklasse *ClearanceSurface*.
Des weiteren können Objekt-Relationen über die Klasse *ObjectRelation* auch auf Teilflächen-Ebene repräsentiert werden. Die Klasse *ObjectRelation* referenziert das Relationsobjekt über seine ID und spezifiziert die Art der Beziehung zwischen den Objekten. Diese Modellierung ist der Enkodierung im CityGML 3.0-Schema nachempfunden. Zusätzlich bietet das Attribut ”adjacentObjects” die Möglichkeit, räumlich-topologisch adjazente Objekte zu referenzieren.
Die ”parent”-Assoziation enthält eine Referenz des Container-Objekts, von welchem die Teilfläche Bestandteil ist.

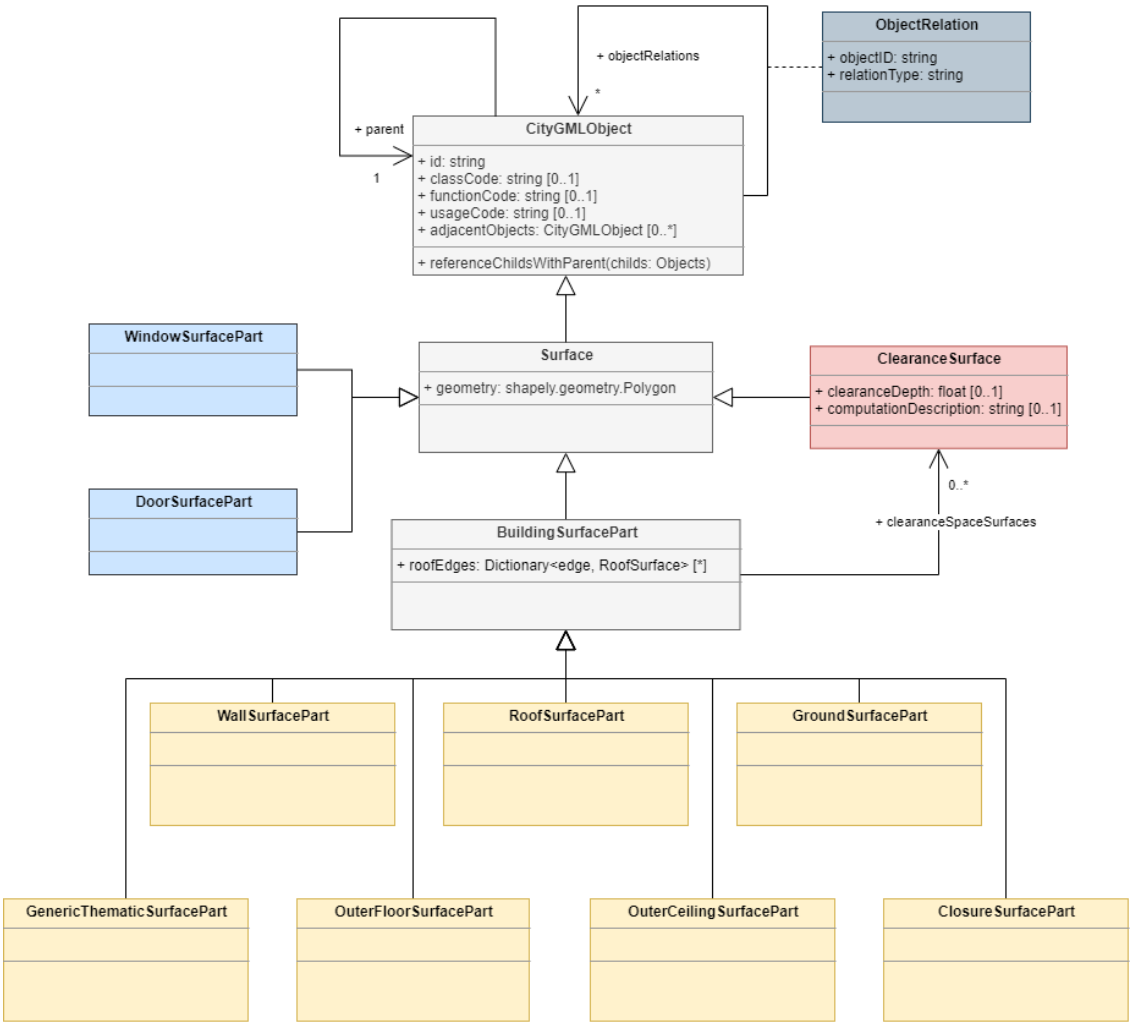


Abbildung 58 Klassendiagramm für die Programm-interne Repräsentation der thematischen Teilflächenklassen.

Thematische Container-Klassen

Die thematischen Container-Klassen aggregieren die (thematischen) Teilflächen. Diese Aggregation der Teilflächen definiert die Multi-Flächengeometrie des Container-Objekts. Die Container-Klassen korrespondieren mit den thematischen Begrenzungsflächen für Gebäude aus dem CityGML-Standard.

Jede thematische Container-Klasse kann für die Modellierung von Gebäudeinstallationen (*BuildingInstallation*) verwendet werden. Auch das ist konform zum CityGML-Standard.

Relationen zu anderen Objekten können über die Klassen *ObjectRelation* hergestellt werden. Das Attribut ”adjacentObjects” bietet ebenfalls die Möglichkeit räumlich-topologisch adjazente Objekte zu referenzieren. Das Elternobjekt wird über die Assoziation ”parent” hergestellt und verweist auf ein Objekt der Klasse *BuildingInstallation, BuildingPart* oder *Building*.

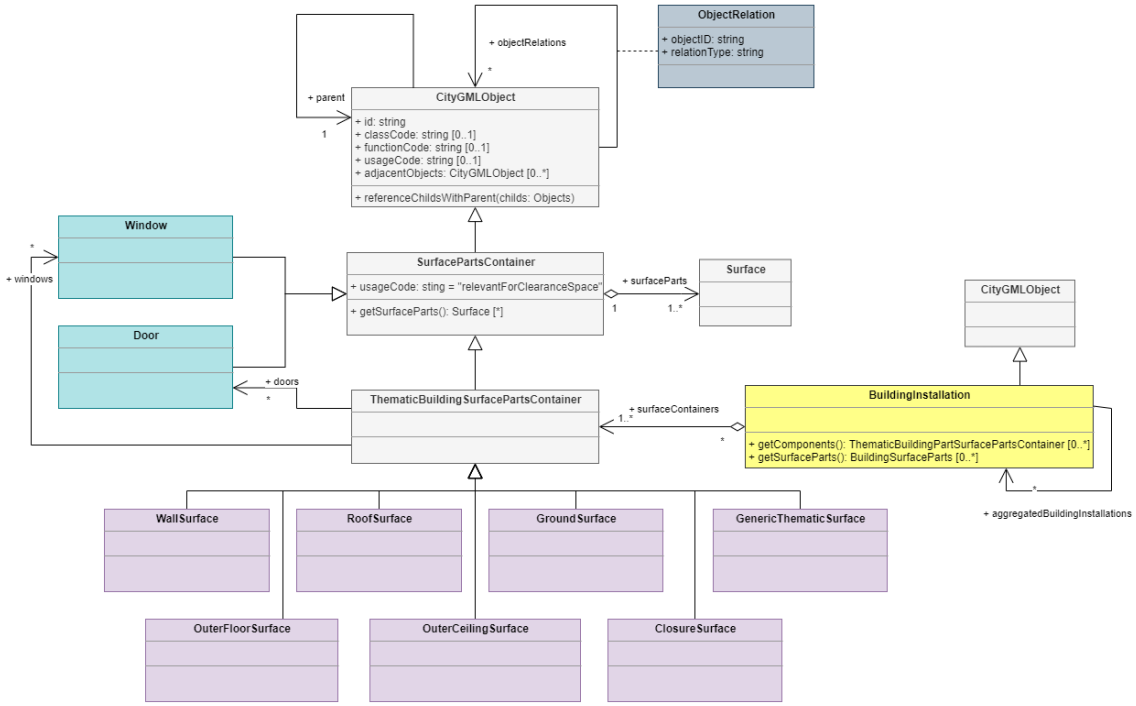


Abbildung 59 Klassendiagramm für die Programm-interne Repräsentation der Container-Klassen.

Gebäude und Gebäudeteil

Ein Objekt der Klasse *Building* bildet die Aggregation von Objekten der thematischen Container-Klassen, sowie Gebäudeinstallationen. Zusätzlich kann es über das Attribut ”buildingParts” auf Instanzen der Klasse *BuildingPart* verweisen, welches als Unterklasse von *Building* wiederum alle Assoziationen und Attribute der *Building*-Klasse erbt. Um die resultierende Abstandsfläche zu erhalten, wird die Operation *buildingClearanceSpace* definiert. Diese berechnet eine Vereinigung aller Abstandsflächen der Gebäudekomponenten in der x-y-Ebene und speichert diese Abstandsfläche im Attribut ”clearanceSpace”.

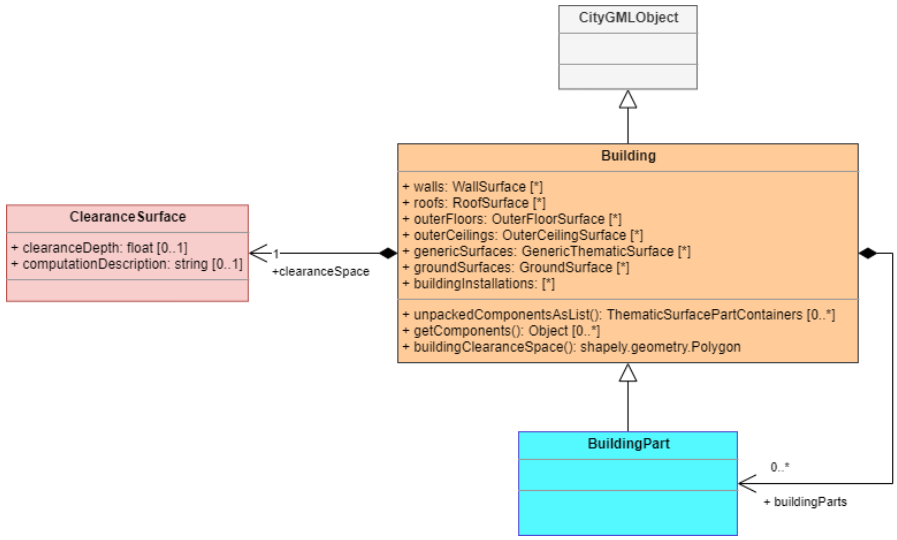


Abbildung 60 Klassendiagramm für die Programm-interne Repräsentation der Klassen für Gebäude und Gebäudeteile.

Flächen aus dem Bebauungsplan/Grundstück des Gebäudes

Die für die Berechnung der Abstandsflächen notwendigen Baugebietsflächen der Klasse *SpatialPlanDevelopmentArea* werden in der Klasse *SpatialPlan* zusammengefasst. Jede Baugebietsfläche enthält ein Attribut ”areaType”, dass die Art der Baugebietsfläche spezifiziert. Die Art der Baugebietsfläche muss in den Schlüsseln (*keys*) des *dictionary*-Attributs ”developmentAreaFactors” enthalten sein. Das *dictionary*-Attribut definiert die Anpassungsfaktoren für die Abstandsflächentiefe, je nach Baugebiet.
Für die Repräsentation des Grundstücks steht die Klasse *PropertyParcel* zur Verfügung.

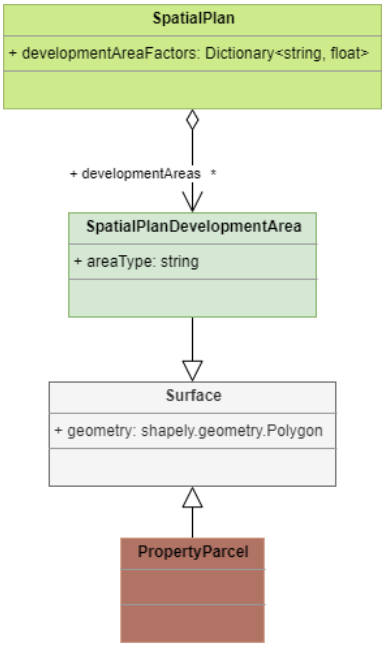


Abbildung 61 Klassendiagramm für die Programm-interne Repräsentation der Klassen für Baugebietsflächen und des Gebäudegrundstücks.

Graph-Struktur des Gebäude-Objekts

Durch die Bereitstellung der Attribute ”adjacentObjects” und ”objectRelations” können sich die Komponenten des Gebäudemodell innerhalb des Gebäudemodells referenzieren. Dies schafft die Möglichkeit, eine Art Graph-Struktur des Gebäudes zu erstellen. Können räumliche Adjazenzen (es erfordert eine Definition, wann Objekte als adjazent bezeichnet werden; bspw. kann definiert werden, dass ab einer 1D (ein-dimensional)-Schnittmenge (oder höher) zweier Objekte (Kantenberührung) eine Adjazenz vorliegt) zwischen Gebäudekomponenten identifiziert werden, so können sich die Gebäudekomponenten untereinander referenzieren, was einer bi-direktionalen Graph-Darstellung des Gebäudes durch dessen Komponenten entspricht.
Es ist notwendig, die topologischen Beziehungen der Gebäudekomponenten untereinander nutzen zu können, da bei der Prüfung einer Gebäudekomponente auf Abstandsflächen oft die Adjazenz zu anderen Gebäudekomponenten berücksichtigt werden muss (siehe Unterabschnitt 4.1.1).

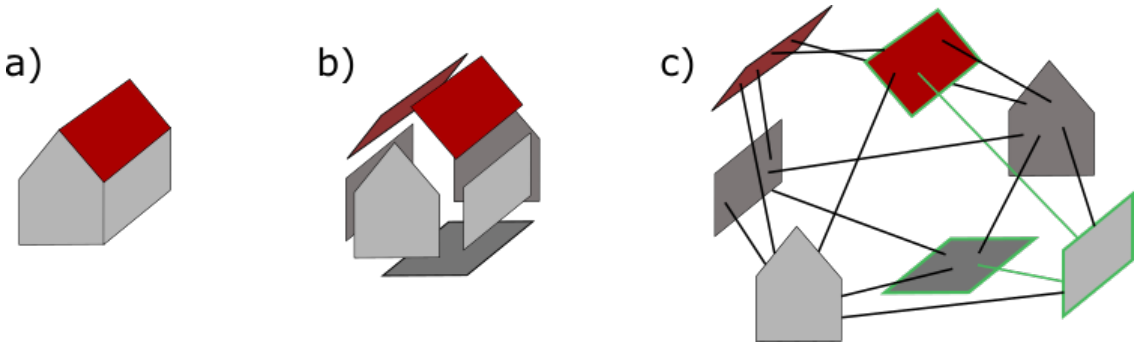


Abbildung 62 Dekomposition eines Gebäudemodells (*a)*) in seine Komponenten (*b)*) und Erstellung eines Komponentengraphs (*c)*). In grün ist beispielhaft ein Durchlauf durch den Graph von einer Dachkomponente zur Bodenkomponente (oder andersherum) dargestellt.

Referenz zur Python-Dokumentation

- CityObject Modul: Python-Dok. Seite 1 - 8

4.2.4 Prozess - Vorverarbeitung

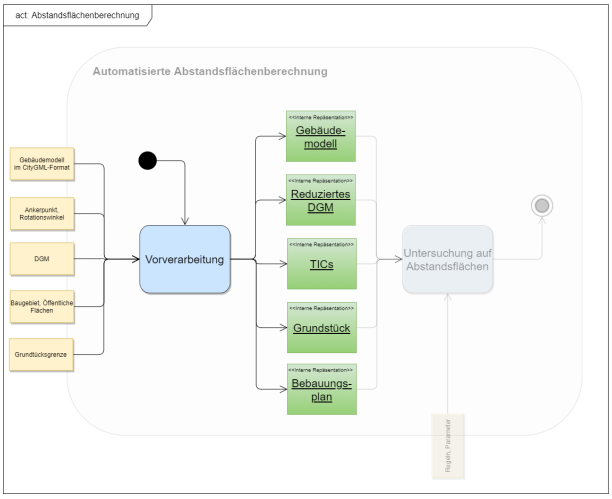


Abbildung 63 Kontext des Prozesses *Vorverarbeitung* innerhalb der Aktivität.

Nachdem die Modellstruktur des eingehenden Gebäudemodells festgelegt wurde und ein valides Dokument für die Verarbeitung bereitsteht, können die Eingangsdaten im Prozess *Vorverarbeitung* eingelesen und für die Abstandsflächenberechnung vorbereitet werden. Im folgenden werden die Schritte innerhalb des ersten Hauptprozesses *Vorverarbeitung* beschrieben.
Die Parameter *Gebäudemodell im CityGML-Format* und *Ankerpunkt, Rotationswinkel* werden aus externen Dateien in das Programm eingelesen. Das Format der Transformationsparameter-Datei hält sich an die Spezifikationen in
Abschnitt 4.1.2. Die Parameter werden über einen *Reader* aus der Datei extrahiert und in globalen Transformationsvariablen gespeichert.
Das Gebäudemodell liegt nach dem Einlesen zunächst als DOM vor.

Der übergeordnete Prozess *Vorverarbeitung* enthält weitere Unterprozesse, welche in Abbildung 64 dargestellt sind.

Im Prozess *Extraktion relevanter Gebäudekomponenten + Georeferenzierung* wird das Gebäude von der DOM-Repräsentation auf die Programm-interne Darstellung des Gebäudemodells abgebildet und simultan mittels der Transformationsparameter georeferenziert. Bei der Extraktion der CityGML-Gebäudekomponenten aus dem DOM wird nur nach den in Abschnitt 4.1.2 und Abschnitt 4.2.3 berücksichtigten Komponenten gefiltert.

Anhand des Gebäudemodells werden verschiedene weitere Unterprozesse durchgeführt.
Im Prozess *Aggregation adjazenter Instanzen der Klasse BuildingInstallation mit unterschiedlicher Ordnung* wird jede Instanz der Klasse *BuildingInstallation* auf Adjazenz zu anderen Gebäudeinstallationen untersucht. Die für eine Adjaznz notwendigen Schnittmengen zweier Objekte sind 1D (Kantenberührung) und 2D (Flächenberührung). Es wird angenommen, dass sich topologische Relationen auf die Berührung beschränken und sich Objekte nicht schneiden. Werden adjazente *BuildingInstallation*-Instanzen gefunden, so wird deren ”classCode”-Attribut betrachtet. Weisen die Gebäudeinstallationen unterschiedliche Ordnungen auf (bspw. die eine ”primaryElement”, die andere ”secondaryElement”), so wird das untergeordnete Element (”secondaryElement”) dereferenziert und dem übergeordnetem Element (”primaryElement”) zugewiesen (Speicherung im Attribut ”aggregatedBuildingInstallation”).

Anschließend findet im Prozess *Initialisierung der topologischen Beziehungen zwischen Gebäudekomponenten* eine generelle Untersuchung auf Adjazenzen zu anderen Gebäudekomponenten statt. Das Resultat ist eine Graph-ähnliche Struktur, da adjazente Gebäudekomponenten im Attribut ”adjacentObjects” gelistet werden, und damit rekursiv von einem Objekt zu all seinen Nachbarn traversiert werden kann (vgl. Abbildung 4.2.3).

Mit Hilfe des Gebäudemodells wird im Unterprozess *Berechnung eines Buffers um Gebäudeprojektion auf horizontale Ebene* ein planarer Buffer in der x-y-Ebene um das Gebäudemodell erzeugt, um im folgenden Prozess *Extraktion aus DGM* nur die Punkte des DGMs zu behalten, die innerhalb dieses Buffers liegen.

Der Prozess *Berechnung der Geländeschnittlinien* erwartet als Eingangsdaten das reduzierte DGM aus dem Prozess *Extraktion aus dem DGM* und das interne Gebäudemodell. Es werden die Schnittlinien der Gebäudeflächen mit dem Gelände berechnet.

Die eingehende Grundstücksgrenze wird auf die interne Darstellung abgebildet. Ebenso werden aus den Flächen des Bebauungsplans eine Bebauungsplan-Instanz entsprechend des internen Datenmodells erzeugt.

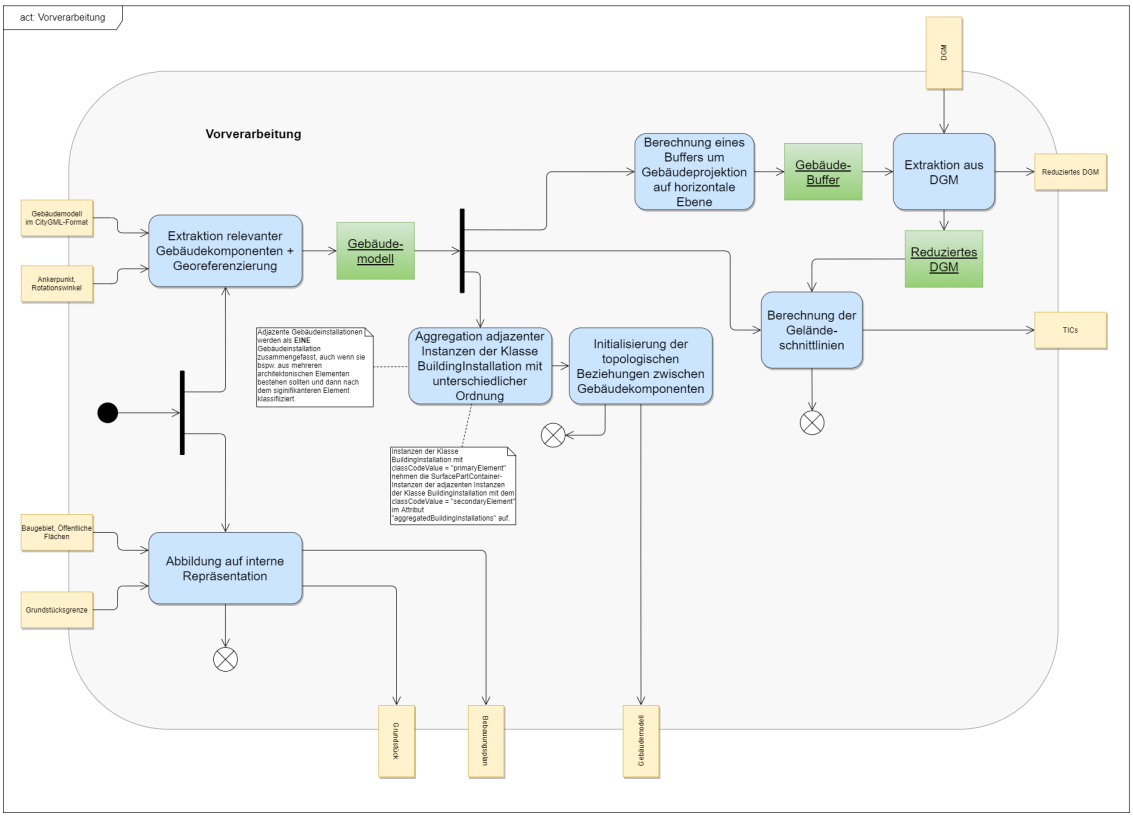


Abbildung 64 Aktivitätsdiagramm des Prozesses *Vorverarbeitung*.

Referenz zur Python-Dokumentation

- *Extraktion relevanter Gebäudekomponenten + Georeferenzierung*:
  - CityGMLProcessing Modul: *CityGMLProcessing*.extractElement, Python-Dok. Seite 10
  - WorldFileReader Modul: Python-Dok. Seite 10
- *Aggregation adjazenter Instanzen der Klasse BuildingInstallation mit unterschiedlicher Ordnung*:
  - GeometricAnalysis Modul: *GeometricAnalysis*.aggregateBuildingInstallations, Python-Dok. Seite 21
- *Initialisierung der topologischen Beziehungen zwischen Gebäudekomponenten*:
  - GeometricAnalysis Modul: *GeometricAnalysis*.createTopologies, Pyhton-Dok. Seite 23
- *Berechnung eines Buffers um Gebäudeprojektion auf horizontale Ebene*:
  - GeometricAnalysis Modul: *GeometricAnalysis*.createDTMBuffer, Pyhton-Dok. Seite 23
- *Extraktion aus DTM*:
  - DTM Modul: *DTM*.readDTM, Pyhton-Dok. Seite 19

- *Berechnung der Geländeschnittlinien*:
  - GeometricAnalysis Modul: GeometricAnalysis.terrainIntersectionCurve, Pyhton-Dok. Seite 36

4.2.5 Prozess - Untersuchung auf Abstandsflächen

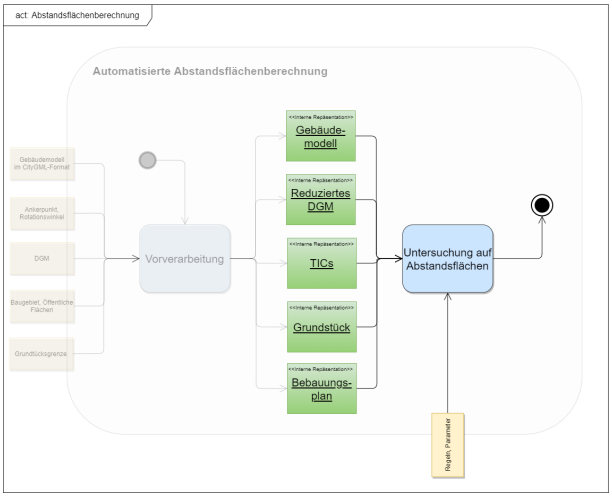


Abbildung 65 Kontext des Prozesses *Untersuchung auf Abstandsflächen* innerhalb der Aktivität.

Aus dem vorherigen Prozess *Vorverarbeitung* gehen folgende Objekte hervor:

- Interne Repräsentationen des Gebäudemodells, des Bebauungsplans und des Grundstücks
- Reduziertes DGM
- TICs

Diese Objekte gehen nun in den Prozess *Untersuchung auf Abstandsflächen* ein. Zunächst werden die Gebäudekomponenten (thematischen Begrenzungsflächen, Gebäudeinstallationen und Gebäudeteile) des internen Gebäudemodells extrahiert. Anschließend folgt eine Iteration über die Gebäudekomponenten, bei der jede Komponente im Unterprozess *Prüfung, ob Abstandsfläche erforderlich* darauf untersucht wird, ob eine Berechnung von Abstandsflächen durchgeführt werden muss, oder nicht. Hierfür gehen als Eingangsparameter sowohl das Grundstück, als auch die Berechnungsregeln mit Parametern ein.
Wird festgestellt, dass eine Abstandsfläche notwendig ist, werden die thematischen Teilflächen der Gebäudekomponente extrahiert. Für jede der Teilflächen wird dann im Prozess *Berechnung der Abstandsfläche* die Abstandsflächenberechnung durchgeführt und die Abstandsfläche mit der zugehörigen Teilfläche referenziert.

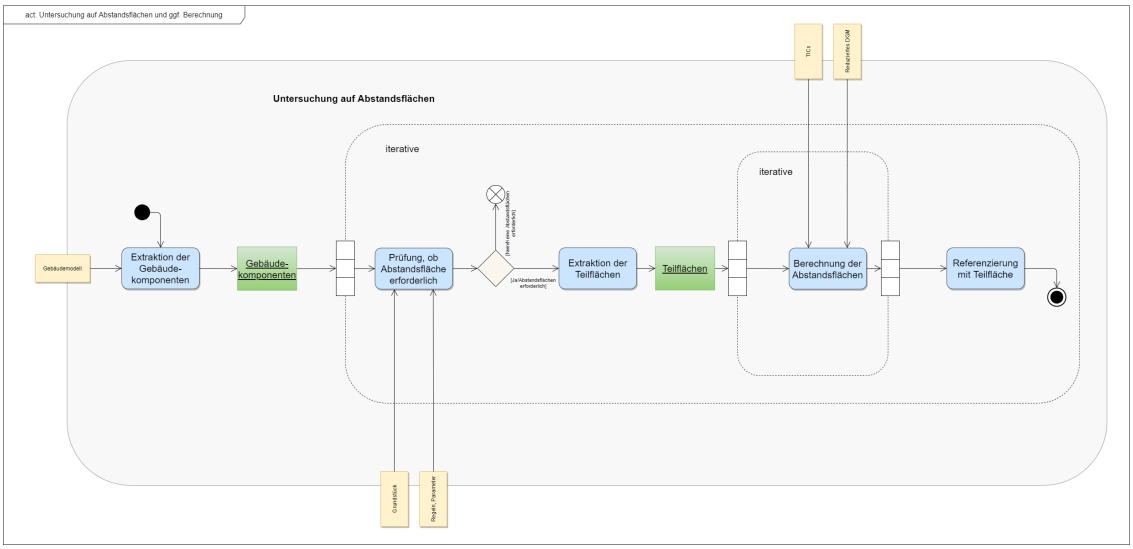


Abbildung 66 Aktivitätsdiagramm des Prozesses *Untersuchung auf Abstandsflächen*.

Referenz zur Python-Dokumentation

- *Untersuchung auf Abstandsflächen*:
  - ClearanceSpaceInspector Modul: *ClearanceSpaceInspector*.inspectBuilding4ClearanceSpaces, Python-Dok. Seite 16

4.2.6 Unterprozess - Prüfung, ob Abstandsfläche erforderlich

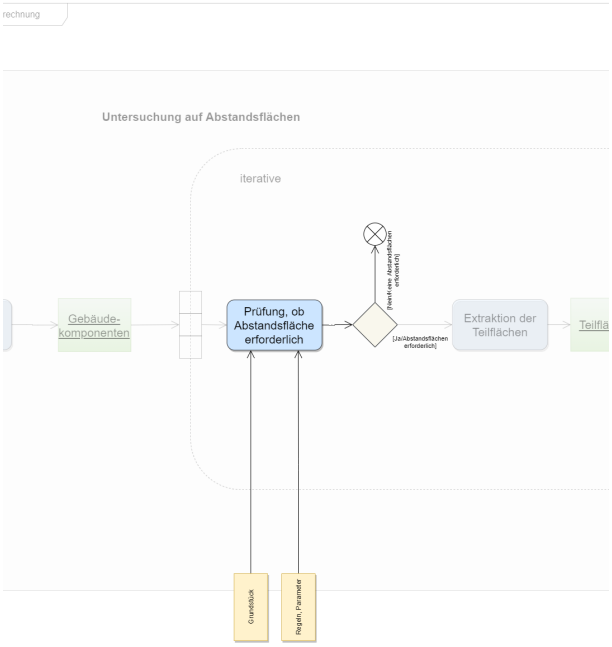


Abbildung 67 Kontext des Prozesses *Prüfung, ob Abstandsfläche erforderlich* innerhalb der Aktivität.

Das Eingangsobjekt für diesen Prozess ist die aktuelle Gebäudekomponente der äußeren Iteration des Prozesses *Untersung auf Abstandsflächen*. Zunächst wird überprüft, ob das Objekt eine Instanz der Klasse *BuildingPart* ist. Ist dies der Fall, wird der Prozess *Untersuchung auf Abstandsflächen* mit dem Gebäudeteil als Eingangsobjekt rekursiv aufgerufen.
Ist das Objekt nicht vom Typ *BuildingPart*, wird im nächsten Schritt weiter geprüft, ob es sich um eine *BuildingInstallation*-Instanz handelt. Trifft dies nicht zu, so ist die aktuelle Komponente eine thematische Gebäudefläche. Es wird verifiziert, ob es sich um eine privilegierte Komponente handelt, indem das Attribut ”usage” ausgewertet wird. Weist es nicht den Wert ”relevantForClearanceSpace” auf, so wird keine Abstandsflächenberechnung durchgeführt, andernfalls schon.
Ist das aktuelle Objekt jedoch vom Type *BuildingInstallation*, so folgen eine Reihe weiterer Tests bzgl. der Art und Funktion der Gebäudeinstallation anhand des Attributs ”function”.
Die Klassifikation der Bauteile hält sich dabei an die in Abschnitt 4.1.1 definierten Typen von Bauteilen, die bei der Abstandsflächenberechnung unberücksichtigt bleiben. Es wird demnach geprüft, ob es sich bei der aktuellen Komponente um ein vortretendes Bauteil (z.B. ein Gesims) handelt (*function = 'structuralElement'*), oder nicht. Wenn es zutrifft, wird anhand der Grenzwerte verifiziert, ob das Bauteil zu weit aus dem Referenzobjekt hervortritt. Ist dies der Fall, sind Abstandsflächen nötig und es findet eine Rekursion mit der Gebäudeinstallation als Eingangsobjekt im Prozess *Untersuchung auf Abstandsflächen* statt.
Handelt es sich nicht um eine hervortretendes Bauteil, wird untersucht, ob die Gebäudeinstallation vom Typ *Dachgaube* oder *Dachaufbau* ist (*function = 'roofConstruction'*). Wenn ja, wird anhand der Lage der Dachkonstruktion zum Grundstück, sowie der Eigenschaften bzgl. der adjazenten Dachkomponente entschieden, ob aus der Konfiguration der Dachkonstruktion Abstandsflächen hervorgehen (für die Beschreibung der Konfiguration siehe Abschnitt 4.1.1). Auch hier kommt es folglich zum rekursiven Aufruf des Prozesses *Untersuchung auf Abstandsflächen*.
Wird die aktuelle Gebäudeinstallation auch nicht als Dachkonstruktion identifiziert, wird als nächstes getestet, ob sie vom Typ *Vorbau* ist (*function = 'porch'*). Der Vorbau wird anhand der in Abschnitt 4.1.1 für Vorbauten beschriebene Konfiguration analysiert und bei Feststellung der Überschreitung der festgelegten Grenzwerte der rekursive Aufruf des Prozesses *Untersuchung auf Abstandsflächen* mit der aktuellen Gebäudeinstallation als Eingangsobjekt getätigt.

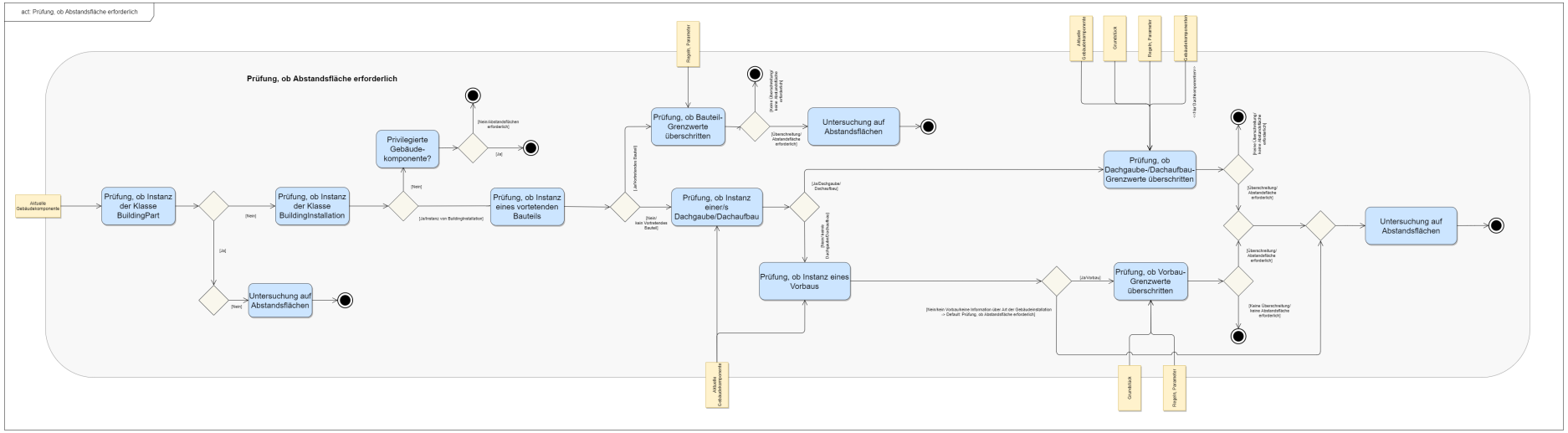


Abbildung 68 Aktivitätsdiagramm des Prozesses *Prüfung, ob Abstandsfläche erforderlich*.

Darstellung des Prozesses als Entscheidungsbaum

Anhand Abbildung 68 ist zu sehen, dass die Prozessfolge entlang von Entscheidungspfaden definiert wird und eine Baumgraph-Struktur aufweist. Der Prozess *Prüfung, ob Abstandsfläche erforderlich* wird deshalb als Entscheidungsbaum dargestellt.

Wie in Abbildung 69 zu sehen, besteht dieser aus einem *root*-Knoten als Wurzelelement. Dieser enthält u.a. ein *operation*-Element, als auch ein *parameter*-Element. Das *operation*-Element ist auf die Werte einer anwendungsspezifischen Auswahl an Operationsbezeichnern beschränkt, für die das *parameter*-Element optional zusätzliche Parameter in Form einer Zeichenkette bereitstellt. Das Ergebnis der Operation dient als Entscheidung, welcher Pfad weiterverfolgt wird. Wird als Rückgabewert der boolsche Wert wahr (*true*) erhalten, wird der Sub-Baumgraph des *true*-Knotens weiter bearbeitet, andernfalls der, des *false*-Knotens. In jedem der betrachteten Knoten dient die Operation der Wahl des nächsten Knotens und dessen Sub-Baumgraph. Das geschieht solange, bis ein Blatt-Element erreicht wird. Dann ist das Resultat der Operation der endgültig Wert des Durchlaufs durch den Entscheidungsbaum.

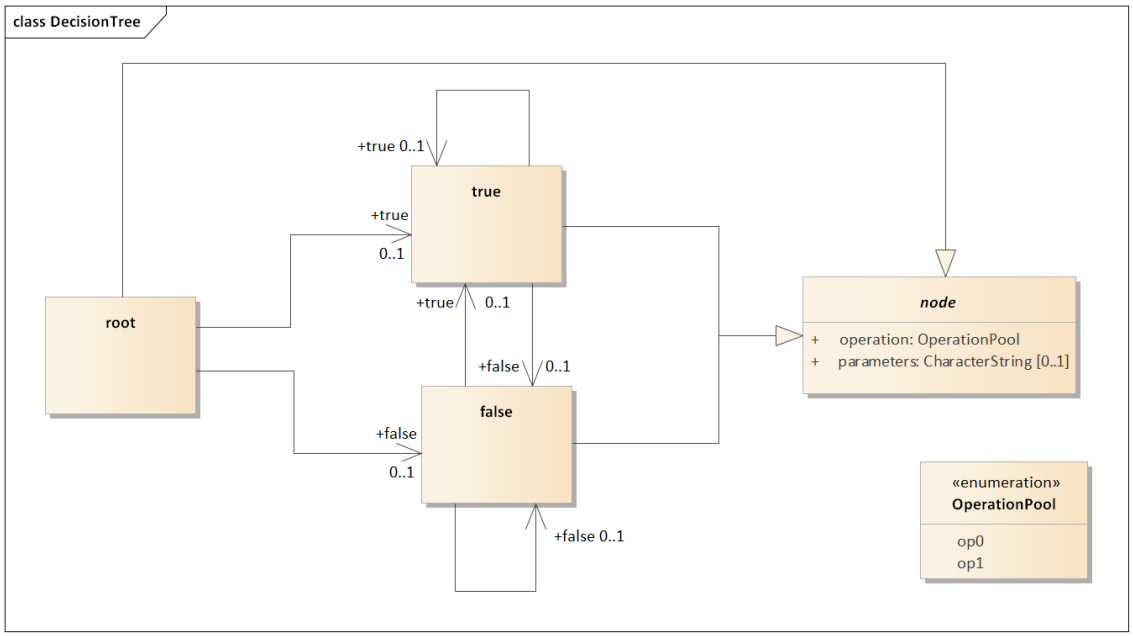


Abbildung 69 UML-Klassendiagramm des Entscheidungsbaums.

Implementierung des Prozesses mittels Entscheidungsbaum

Die Enkodierung des Entscheidungsbaums erfolgt im XML-Format. Dies bietet sich an, da XML-Dokumente als DOM in den Programmspeicher eingelesen werden können und damit bereits in der hierarchischen Baumstruktur vorliegen.

Für die Prozessierung des Entscheidungsbaums in Python wird eine Art Regel-Maschine verwendet. Dieser wird ein *Dictionary* übergeben, das die Namen aus dem *OperationPool* (siehe Listing 4.5) des Entscheidungsbaum-Schemas als *Keys* enthält. Die entsprechenden Werte sind Funktionen, die mit den jeweiligen Prozessen im Aktivitätsdiagramm (siehe Abbildung 68) korrespondieren. Über *function binding* wurden sie an alle nicht als Zeichenketten darstellbaren Parameter gebunden, sodass die Funktionen bei ihrer Ausführung schon alle nötigen Parameter in sich tragen.

Bei der Prozessierung des Entscheidungsbaums extrahiert die Regel-Maschine aus dem aktuellen Knoten den Namen der auszuführenden Operation. Die Funktion des *Dictionary*-Eintrags mit dem *Key* des Operationsnamen wird aufgerufen und basierend auf dem Rückgabewert der Subgraph des nächsten *true*- oder *false*-Knotens ausgewertet.
In den Fällen, in denen der Prozess *Prüfung, ob Abstandsfläche erforderlich* rekursiv aufgerufen wird, startet die Regel-Maschine den Durchlauf durch den Entscheidungsbaum mit der zu überprüfenden Gebäudekomponente neu.

In Listing 4.5 ist ein Schema zur Validierung des Entscheidungsbaums dargestellt. Die Enumeration, welche die Namen der Operationen beinhaltet, muss anwendungsspezifisch angepasst werden und ist in dem Schema mit Platzhaltern versehen.
In Listing 4.6 ist der Entscheidungsbaum korrespondierend zum Aktivitätsdiagramm in Abbildung 68 im XML-Format zu sehen.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="false" type="false"/>
  <xs:complexType name="false">
    <xs:complexContent>
      <xs:extension base="node">
        <xs:sequence>
          <xs:element name="false" type="false" minOccurs="0" maxOccurs="1"/>
          <xs:element name="true" type="true" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="node" type="node"/>
  <xs:complexType name="node" abstract="true">
    <xs:sequence>
      <xs:element name="operation" type="OperationPool" minOccurs="1" maxOccurs="1"/>
      <xs:element name="parameters" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="OperationPool">
    <xs:restriction base="xs:string">
      <xs:enumeration value="op0"/>
      <xs:enumeration value="op1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="root" type="root"/>
  <xs:complexType name="root">
    <xs:complexContent>
      <xs:extension base="node">
        <xs:sequence>
          <xs:element name="false" type="false" minOccurs="0" maxOccurs="1"/>
          <xs:element name="true" type="true" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="true" type="true"/>
  <xs:complexType name="true">
    <xs:complexContent>
      <xs:extension base="node">
        <xs:sequence>
          <xs:element name="false" type="false" minOccurs="0" maxOccurs="1"/>
          <xs:element name="true" type="true" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Listing 4.5: decisiontree.xsd

```xml
<?xml version="1.0" encoding="utf-8"?>
<root>
  <operation>isBuildingPart</operation>

  <true>
    <!-- Recursive call -->
    <operation>clearanceSpaceInspection</operation>
  </true>

  <false>
    <operation>isBuildingInstallation</operation>
    <!-- <parameters></parameters> -->

    <!-- path: is a building installation -->
    <true>
      <operation>isStructuralElement</operation>
      <true>
        <operation>exceedsStructuralElementThresholds</operation>
        <true>
          <operation>clearanceSpaceInspection</operation>
          <!-- recursive call of the inspection function -->
        </true>
      </true>

      <false>
        <operation>isRoofConstruction</operation>
        <true>
          <operation>exceedsRoofConstructionThresholds</operation>
          <true>
            <operation>clearanceSpaceInspection</operation>
          </true>
        </true>

        <false>
          <operation>isPorch</operation>
          <true>
            <operation>exceedsPorchThresholds</operation>
            <true>
              <operation>clearanceSpaceInspection</operation>
            </true>
          </true>
          <!-- default operation -->
          <false>
            <operation>clearanceSpaceInspection</operation>
          </false>
        </false>
      </false>
    </true>

    <!-- path: is not a building installation -->
    <false>
      <operation>isNotPrivilegedComponent</operation>
      <!-- there is neither a 'true' node nor a 'false' node, so return the decision -->
      <!-- if it is not a relevant component 'false' is returned -->
    </false>
  </false>

</root>
```

Listing 4.6: decisionTree.xml

Referenz zur Python-Dokumentation

- *Prüfung, ob Abstandsfläche erforderlich*:
  - ClearanceSpaceInspector Modul: *ClearanceSpaceInspector*.recursiveInspection, Python-Dok. Seite 16
  - ClearanceSpaceInspector Modul: *ClearanceSpaceInspector*.calculateClearanceSpace, Python-Dok. Seite 16
  - RuleEngine Modul: *RuleEngine*.startRuleEngine, Python-Dok. Seite 40

- GeometricAnalysis Modul: *GeometricAnalysis*.isRelevantSurfacePart, Python-Dok. Seite 29
- GeometricAnalysis Modul: *GeometricAnalysis*.privilegeBuildingInstallation, Python-Dok. Seite 33
- GeometricAnalysis Modul: *GeometricAnalysis*.privilegeBuildingPart, Python-Dok. Seite 33
- GeometricAnalysis Modul: *GeometricAnalysis*.privilegeComponent, Python-Dok. Seite 34

4.2.7 Unterprozess - Berechnung der Abstandsflächen

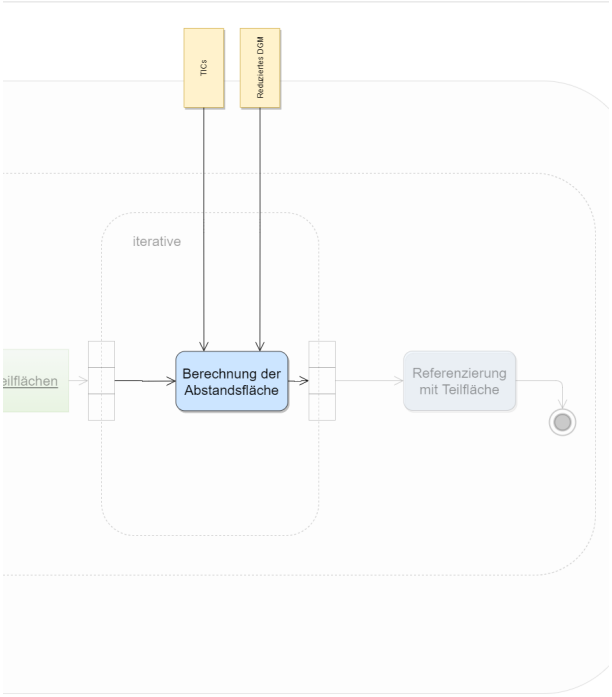


Abbildung 70 Kontext des Prozesses *Berechnung der Abstandsflächen* innerhalb der Aktivität.

Wird in dem Prozess *Prüfung, ob Abstandsfläche erforderlich* festgestellt, dass für die aktuelle Gebäudekomponente vom Typ einer thematischen Gebäudefläche eine Abstandsfläche erforderlich ist, werden zunächst deren Teilflächen extrahiert. Jede der Teilflächen geht anschließend in den Prozess *Berechnung der Abstandsflächen* ein.
Die prinzipielle Abstandsflächenberechnung findet dann je Liniensegment einer Teilfläche statt, d.h. jedes Liniensegment resultiert in seiner eigenen Abstandsfläche.

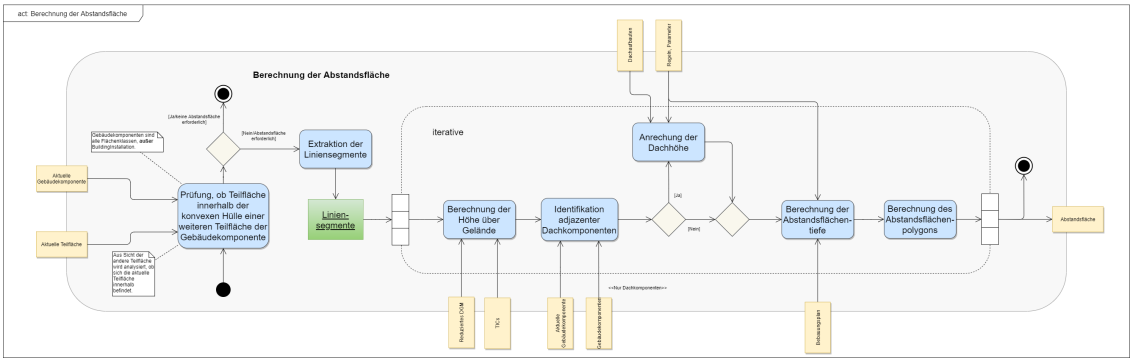


Abbildung 71 Aktivitätsdiagramm des Prozesses *Berechnung der Abstandsflächen*.

Im ersten Prozess wird inspiziert, ob sich die aktuelle Teilfläche innerhalb einer konvexen Hülle einer nach unten, oder zur Seite konkaven anderen Teilfläche der aktuellen Gebäudekomponente befindet. Stellt es sich als wahr heraus, muss für die aktuelle Teilfläche keine Abstandsfläche berechnet werden. Da sowohl die horizontalen als auch vertikalen Koordinaten der enthaltenen Teilfläche innerhalb der konvexen Hülle der anderen Teilfläche liegen, ist damit auch die Abstandsfläche der aktuellen Teilfläche in der Abstandsfläche der anderen Teilfläche enthalten.
Die entsprechende Konfiguration ist in Abbildung 72 dargestellt.
Es gilt für alle Liniensegmente $S = (s_n)_{n=0}^{N} = (s_0, s_1, ...)$ der aktuellen Teilfläche A (orange), dass eine zusammenhängende Liniensegmentsequenz $L = (l_k)_{k=0}^{K} = (l_0, l_1, ...)$ von mindestens einem Liniensegment der Liniensegmente J der anderen Teilfläche B (grün) existieren muss, dessen untere Grenze des von L abgedeckten Intervalls $I_h^B = [q, w]$ genau im Bereich $I_u^A = [a, b]$ über der oberen Grenze des Intervalls $I_h^A = [c, d]$ liegt.

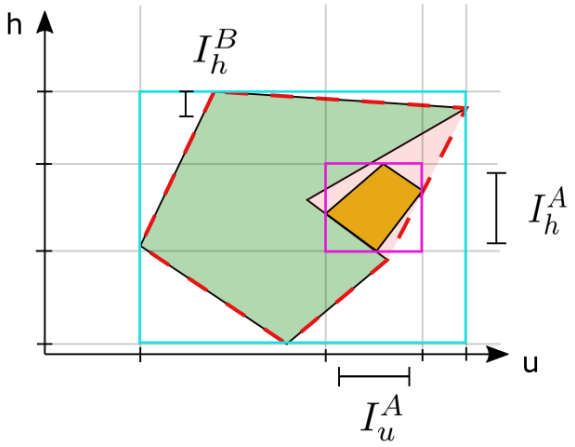


Abbildung 72 Aktuelle Teilfläche (orange) innerhalb der konvexen Hülle (rot) einer anderen zur Seite geöffneten konkaven Teilfläche (grün).

Das ist allerdings nicht der Fall, wenn die Teilfläche innerhalb einer konvexen Hülle einer nach oben hin konkaven anderen Teilfläche ist (siehe Abbildung 73). Dann ergibt sich durch die größeren vertikalen Koordinaten im Bereich I_u^A eine größere Abstandsflächentiefe für Fläche A (orange) als für Fläche B, da diese als Eingangsparameter die Höhe (hier die vertikale Koordinate) erwartet.

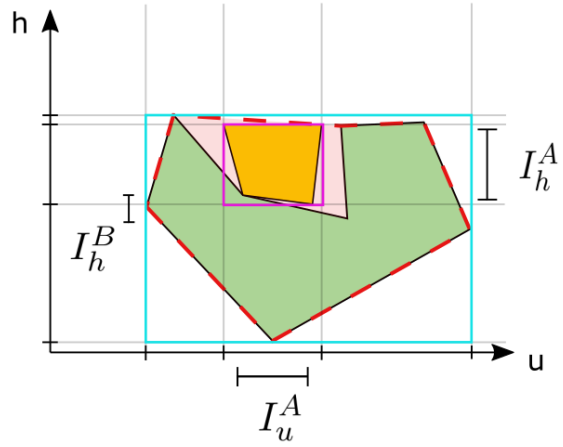


Abbildung 73 Aktuelle Teilfläche (orange) innerhalb der konvexen Hülle (rot) einer anderen nach oben geöffneten konkaven Teilfläche (grün).

Ist die Konfiguration in Abbildung 73 der Fall, oder befindet sich die aktuelle Teilfläche in keiner konvexen Hülle einer anderen Teilfläche, wird für jedes Liniensegment der aktuellen Teilfläche, sofern eine horizontale Komponenten im Richtungsvektor zwischen Start- und Endpunkt ungleich 0 existiert (würde in einem Polygon mit einer Breite von 0 resultieren), die Höhe über Gelände berechnet.
Wenn die aktuelle Teilfläche Adjazenz zu einer Dachfläche aufweist, wird die Höhe der Dachfläche mit einem Faktor, der je nach Dachkonfiguration variiert (definiert in der BauO) an die zuvor berechnete Höhe über Gelände hinzuaddiert.
Anschließend wird die berechnete Höhe mit einem Faktor entsprechend des Bebauungsplans angepasst. Das Resultat entspricht der Abstandsflächentiefe T. Die Konstruktion der Abstandsfläche mittels Liniensegment, sowie der Abstandsfläche T ist in Abbildung 74 dargestellt. Mittels Startpunkt (P_i) und Endpunkt (P_{i+1}) des Liniensegments wird die Länge L sowie der der Abstandsfläche berechnet. Über den mit T skalierten Normalenvektor (in blau dargestellt) werden die Polygon-Punkte der Abstandsfläche erhalten. Da die Abstandsflächen parallel zur x-y-Ebene sein sollen, wird die z-Komponente der Flächennormalen bei geneigten Teilflächen vor der Normalisierung zu 0 gesetzt.
Die Punkte des Abstandsflächenpolygon werden nach der Rechten-Hand-Regel derart geordnet, dass die Flächennormale in positive z-Richtung zeigt.

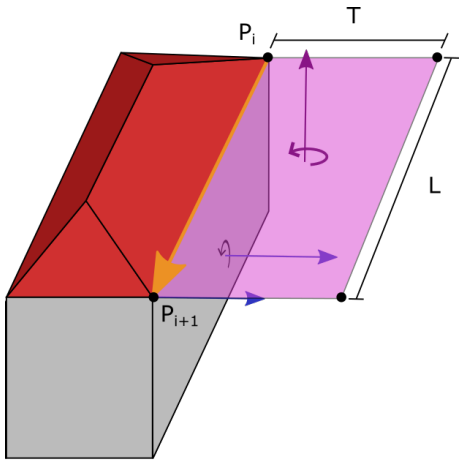


Abbildung 74 Darstellung der Konstruktion der Abstandsflächen für ein Liniensegment einer Teilfläche.

Referenz zur Python-Dokumentation

- *Berechnung der Abstandsflächen*:
  - ClearanceSpaceInspector Modul: *ClearanceSpaceInspector*.calculateClearanceSpace, Python-Dok. Seite 16
  - ClearanceSpaceInspector Modul: *ClearanceSpaceInspector*.clearanceSpace, Python-Dok. Seite 16

4.2.8 Implementierung im Safe Software FME Transformer ”PythonCaller”

In Listing 4.7 ist der Quellcode zu sehen, der die beschriebenen Prozesse implementiert. Start und Ende der beiden Hauptprozesse *Vorverarbeitung* und *Untersuchung auf Abstandsflächen* sind im Code dokumentiert. Für die Implementierung der Unterprozesse wird auf die Python-Dokumentation in Anhang D verwiesen.

```
1  import fme;
2  import fmeobjects;
3
4  import xml.etree.ElementTree as ET;
5  import xml.dom.minidom as XML;
6  from functools import partial;
7
8  from shapely.geometry import Point, Polygon, MultiPolygon;
9
10 import sys;
11 sys.path.append("./python");
12
13 import CityGMLObject as citygmlobj;
14 import CityGMLProcessing as citygml;
15 import DTM as dtm;
16 import GeometricAnalysis as ga;
17 import ClearanceSpaceInspector as csi;
18 import RuleEngine as rEn;
19 import WorldFileReader as wldReader;
20
21 # Template Class Interface:
22 # When using this class, make sure its name is set as the value of
23 # the 'Class or Function to Process Features' transformer parameter
24 spatialPlanFactors = {
25     "öffentlicheVerkehrsfläche": 0.2,
26     "öffentlicheWasserfläche": 0.2,
27     "öffentlicheGrünfläche": 0.2,
```

```
    "Kleinsiedlungsgebiet": 0.4,
    "reinesWohngebiet": 0.4,
    "allgemeinesWohngebiet": 0.4,
    "besonderesWohngebiet": 0.4,
    "Dorfgebiet": 0.4,
    "Mischgebiet": 0.4,
    "urbanesGebiet": 0.4,
    "Kerngebiet": 0.4,
    "Gewerbegebiet": 0.2,
    "Industriegebiet": 0.2,
    "Sondergebiet": 0.4
    };

functionPool_Preprocessing = {
        "getTransformationParameters": wldReader.WorldFileReader.getParameters,
        "setBuildingNamespaceAndCityGMLVersion": citygml.setNamespacesAndCityGMLVersion,
        "extractBuilding": citygml.extractElement,
        "createTopologies": ga.createTopologies,
        "aggregateBuildingInstallations": ga.aggregateBuildingInstallations,
        "createDTMBuffer": ga.createDTMBuffer,
        "terrainIntersectionCurve": ga.terrainIntersectionCurve
        };

class FeatureProcessor(object):
    def __init__(self):
        try:
            # Building terrain intersection curves
            self.buildingTIC = [];

            # Spatial plan instance
            self.spatialPlan = citygmlobj.SpatialPlan([], spatialPlanFactors);

            # Property boundary instance
            self.propertyParcel = None;

            # DTM point list
            self.dtmBuildingDataPoints = [];
            self.dtmBuildingDataPointValues = [];
            self.dtmBuildingPoints = [];

            # DTM buffer
            self.building_XYProjected = None;
            self.buildingDTMBuffer = None;

            # ================================================================================
            #                        Start of process 'Vorverarbeitung'
            # ================================================================================

            # Extract transformation parameters from world file
            print("READING PARAMETERS FROM WORLD FILE");
            worldfilename = "./transformation.wld";
            parameters = functionPool_Preprocessing["getTransformationParameters"](wldReader.WorldFileReader(worldfilename)
      );

            # Set the parameters in the CityGML modul
            citygml.transformationParams = parameters;

            # Extract filename of building model from macro values
            filename = fme.macroValues['building_model'];

            # Parse XML file
            print("PARSING BUILDGIN MODEL FILE: ", filename);
            domObject = XML.parse(filename);

            # PROGRAM WORKS ONLY WITH INLINE PROFILE -> SEE CITYGML 2.0 SPECIFICATION
            functionPool_Preprocessing["setBuildingNamespaceAndCityGMLVersion"](domObject);

            # Extract the building from the DOM
            print("BUILDING EXTRACTION FROM DOM");
            self.building = functionPool_Preprocessing["extractBuilding"](domObject);

            # Unlink the DOM object since the extraction has finished
            domObject.unlink();

            # Analyse the topologies of the building components
            print("CREATING BUILDING COMPONENT TOPOLOGIES");
            functionPool_Preprocessing["createTopologies"](self.building);

            # Aggregate adjacent building installations which are of different priority
            print("AGGREGATING BUILDING INSTALLATIONS");
            functionPool_Preprocessing["aggregateBuildingInstallations"](self.building.buildingInstallations);

```

```
            # Each line segment is projected on the x-y-plane
            print("CREATING DTM BUFFER");
            self.building_XYProjected, self.buildingDTMBuffer = functionPool_Preprocessing["createDTMBuffer"](self.building
    .unpackedComponentsAsList(), bufVal=4.0);
        except:
            print("ERROR IN INIT() FUNCTION");

    def input(self, feature):
        # Check if the feature posses the attribute 'cscObject'
        attributeAbsent = feature.getAttributeNullMissingAndType("cscObject")[2];

        # If the feature does not have the required attribute, do nothing and quit from the function
        if attributeAbsent == fmeobjects.FME_ATTR_UNDEFINED:
            return;

        # Check which value the attribute 'cscOject' has
        cscObjectValue = feature.getAttribute("cscObject");

        if cscObjectValue == "dtmPoint":
            self.__dtmClipping(feature, self.buildingDTMBuffer);
        else:
            # boundaryCurve = feature.getGeometry().getBoundaryAsCurve();
            # Extract coordinates from boundary curve
            pointList = feature.getAllCoordinates();
            print(pointList);
            geometry_tmp = Polygon(pointList);

            if cscObjectValue == "propertyParcel":
                # A closed polygon geometry according to the right hand rule is expected
                if isinstance(feature.getGeometry(), fmeobjects.FMEPolygon) == True:
                    self.propertyParcel = citygmlobj.PropertyParcel("", geometry_tmp);
                    print("PROPERTY BOUNDARY GEOMETRY WAS DEFINED");
            else:
                if cscObjectValue == "developmentArea":
                    # A closed polygon geometry according to the right hand rule is expected
                    if isinstance(feature.getGeometry(), fmeobjects.FMEPolygon) == True:
                        # Extract the area type
                        areaType = feature.getAttribute("developmentAreaType");

                        devArea_tmp = citygmlobj.SpatialPlanDevelopmentArea("", geometry_tmp, areaType);

                        # Add to spatial plan
                        self.spatialPlan.developmentAreas.append(devArea_tmp);
                        print("DEVELOPMENT AREA '{0}' WAS ADDED TO THE SPATIAL PLAN".format(areaType));

    def __dtmClipping(self, fmePoint, clipper):
        # Make a shapely point
        point = Point(fmePoint.getGeometry().getXYZ());

        # If the point lays outside of the clipping geometry return from function and do noting
        if clipper.contains(point) == False:
            return;

        # Store point
        self.dtmBuildingDataPoints.append([point.x, point.y]);
        self.dtmBuildingDataPointValues.append(point.z);
        self.dtmBuildingPoints.append(point);

    def close(self):
        print(self.dtmBuildingPoints);
        # Compute the building terrain intersection curve
        print("COMPUTING THE BUILDING TIC");
        self.buildingTIC = functionPool_Preprocessing["terrainIntersectionCurve"](self.building.unpackedComponentsAsList(),
      self.dtmBuildingPoints);

        # Compose the digital terrain model
        digitalTerrainModel = (self.dtmBuildingDataPoints, self.dtmBuildingDataPointValues);

        # ------------------------------ End of process 'Vorverarbeitung' ------------------------------

        # Initialize CSI
        clearanceSpaceInspector = csi.ClearanceSpaceInspector(self.spatialPlan, self.propertyParcel, self.building,
    digitalTerrainModel, self.buildingTIC);

        # Set parameters
        clearanceSpaceInspector.setClearanceSpaceConfiguration();
        clearanceSpaceInspector.setDefaultAdjustmentFactor();
        clearanceSpaceInspector.setMinimumClearanceSpaceDepth();

        # Set consequence function
        clearanceSpaceInspector.setConsequenceFunction(csi.ClearanceSpaceInspector.calculateClearanceSpace);
```

```
188         # Check if components can be privileged
189         clearanceSpaceInspector.privilegeComponents();
190 
191         # Compose the function set for the rule engine => Key names correspond to operations in decisionTree.xml
192         functionPool_RuleEngine = {
193                         "isBuildingPart": lambda obj: isinstance(obj, citygmlobj.BuildingPart),
194                         "isBuildingInstallation": lambda obj: isinstance(obj, citygmlobj.BuildingInstallation),
195                         "isStructuralElement": lambda obj: obj.functionCode == "structuralElement",
196                         "isRoofConstruction": lambda obj: obj.functionCode == "roofConstruction",
197                         "isPorch": lambda obj: obj.functionCode == "porch",
198                         "isNotPrivilegedComponent": lambda obj: obj.usageCode == "relevantForClearanceSpace",
199                         "isNotOnPropertyBoundary": ga.isNotOnPropertyBoundary,
200                         "exceedsStructuralElementThresholds": partial(ga.genericExceedsThresholds, checkingFunctions=[
    partial(ga.protrusionViolation, rule='protrusion > 1.5')], positiveConsequenceFunc = ga.setAll2Relevant),
201                         "exceedsRoofConstructionThresholds": partial(ga.genericExceedsThresholds, checkingFunctions=[ga.
    roofConstructionViolation]),
202                         "exceedsPorchThresholds": partial(ga.genericExceedsThresholds, checkingFunctions=[ga.
    porchLengthRatioViolation, ga.protrusionViolation,
203                                                                 partial(ga.porchPropertyBoundaryDistanceViolation,
    propertyParcel=self.propertyParcel)]),
204                         "porchLengthRatioViolation": ga.porchLengthRatioViolation,
205                         "exceedsProtrusionThreshold": ga.protrusionViolation,
206                         "roofConstructionViolation": ga.roofConstructionViolation,
207                         "roofConstructionLengthRatio": ga.roofConstructionLengthRatio,
208                         "clearanceSpaceInspection": partial(csi.ClearanceSpaceInspector.recursiveInspection, self=
    clearanceSpaceInspector)
209                       };
210 
211         # Read the descision tree
212         decisionTree = XML.parse('./decisionTree.xml');
213 
214         # Initialize rule engine
215         ruleEngine = rEn.RuleEngine(functionPool_RuleEngine, decisionTree);
216 
217         # Set the rule engine in the CSI
218         clearanceSpaceInspector.setRuleEngine(ruleEngine);
219 
220         # Start clearance space inspection
221         print("STARTING THE CLEARANCE SPACE INSPECTION");
222 
223         # ============================================================================================
224         #                         Start of process 'Untersuchung auf Abstandsflächen'
225         # ============================================================================================
226 
227         clearanceSpaceInspector.inspectBuilding4ClearanceSpaces();
228 
229         # ---------------------- End of process 'Untersuchung auf Abstandsflächen' ----------------------
230 
231         # Create FMEFeatures of the clearance space surfaces (for each of surface parts of the components)
232         print("WRITING OUT THE CLEARANCE SPACE FEATURES");
233         for component in self.building.unpackedComponentsAsList():
234             # Extract the parent id of surface part
235             parentID = component.id;
236             for surfacePart in component.getSurfaceParts():
237                 for clearanceSpaceSurface in surfacePart.clearanceSpaceSurfaces:
238                     fmeFeature = self.__createFMEFeature(clearanceSpaceSurface, parentID);
239 
240                     # Write the feature out
241                     self.pyoutput(fmeFeature);
242 
243 
244         # Create a FMEFeature for the resulting total area of the clearance space
245         buildingClearanceSpaceFMEFeature = self.__createBuildingClearanceSpace();
246 
247         # Write the feature out
248         self.pyoutput(buildingClearanceSpaceFMEFeature);
249 
250         # Create the building TIC feature
251         buildingTICFeature = self.__createBuildingTICFeature(self.buildingTIC);
252 
253         # Write out tic feature
254         self.pyoutput(buildingTICFeature);
255 
256     def __createBuildingTICFeature(self, ticList):
257         # Initialize feature
258         ticFeature = fmeobjects.FMEFeature();
259 
260         # Initialize feature geometry
261         multiCurve = fmeobjects.FMEMultiCurve();
262         print("TICs", ticList);
263         # Iterate over tic list and append lines to multi curve
264         for tic in ticList:
```

```
265             # Extract coordinates from shapely line string
266             points = list(tic.coords);
267             line_tmp = fmeobjects.FMELine(points);
268             multiCurve.appendPart(line_tmp);
269 
270         # Set feature geometry
271         ticFeature.setGeometry(multiCurve);
272 
273         # Set attributes
274         ticFeature.setAttribute("gml_parent_id", self.building.id);
275         ticFeature.setAttribute("clearanceSpaceType", "buildingTIC");
276 
277         return ticFeature;
278 
279     def __createBuildingClearanceSpace(self):
280         # Compute the union of all clearance spaces
281         self.building.buildingClearanceSpace();
282         geometry = self.building.clearanceSpace.geometry;
283 
284         # FMEGeometry
285         fmeGeometry = None;
286 
287         if isinstance(geometry, MultiPolygon) == True:
288             fmeGeometry = fmeobjects.FMEMultiSurface();
289             for g in geometry.geoms:
290                 fmeGeometry.appendPart(fmeobjects.FMEFace(list(g.exterior.coords), fmeobjects.FME_CLOSE_3D_EXTEND_MODE));
291         else:
292             fmeGeometry = fmeobjects.FMEFace(list(geometry.exterior.coords), fmeobjects.FME_CLOSE_3D_EXTEND_MODE);
293 
294         # Create a new FME feature
295         feature = fmeobjects.FMEFeature();
296 
297         # Set the geometry of the FME feature
298         feature.setGeometry(fmeGeometry);
299 
300         # Set the gml_parent_id as attribute to make it applicable for hierachical aggregation in the GML writer
301         feature.setAttribute("gml_parent_id", self.building.id);
302 
303         feature.setAttribute("clearanceSpaceType", "hull");
304 
305         return feature;
306 
307     def __createFMEFeature(self, obj, parent_id):
308         # Extract the geometry from obj
309         geom = obj.geometry;
310         # print(geom);
311 
312         # Create a FME geometry from shapely geometry
313         # print("INITIALISING FEATURE GEOMETRY");
314         fmeFace = fmeobjects.FMEFace(list(geom.exterior.coords), fmeobjects.FME_CLOSE_3D_EXTEND_MODE);
315 
316         # Create a new FME feature
317         # print("INITIALISING FEATURE");
318         feature = fmeobjects.FMEFeature();
319 
320         # Set the geometry of the FME feature
321         feature.setGeometry(fmeFace);
322 
323         # Set the gml_parent_id as attribute to make it applicable for hierachical aggregation in the GML writer
324         feature.setAttribute("gml_parent_id", parent_id);
325 
326         return feature;
```

Listing 4.7: clearanceSpaceComputation.py

4.2.9 Anwendung des Algorithmus zur Abstandsflächenberechnung auf verschiedene Gebäudemodelle

Das in der Python-Dokumentation auf Seite 43 ff. bereitgestellte Skript wird auf verschiedene Testdatensätze angewandt. Ergebnisse sind in Abbildung 75 dargestellt. Die Eingangsgebäudemodelle befinden sich auf der linken Seite, die prozessierten und beispielhaft in ein DGM integrierten Gebäudemodelle mit assoziierten Abstandsflächen auf der rechten Seite.

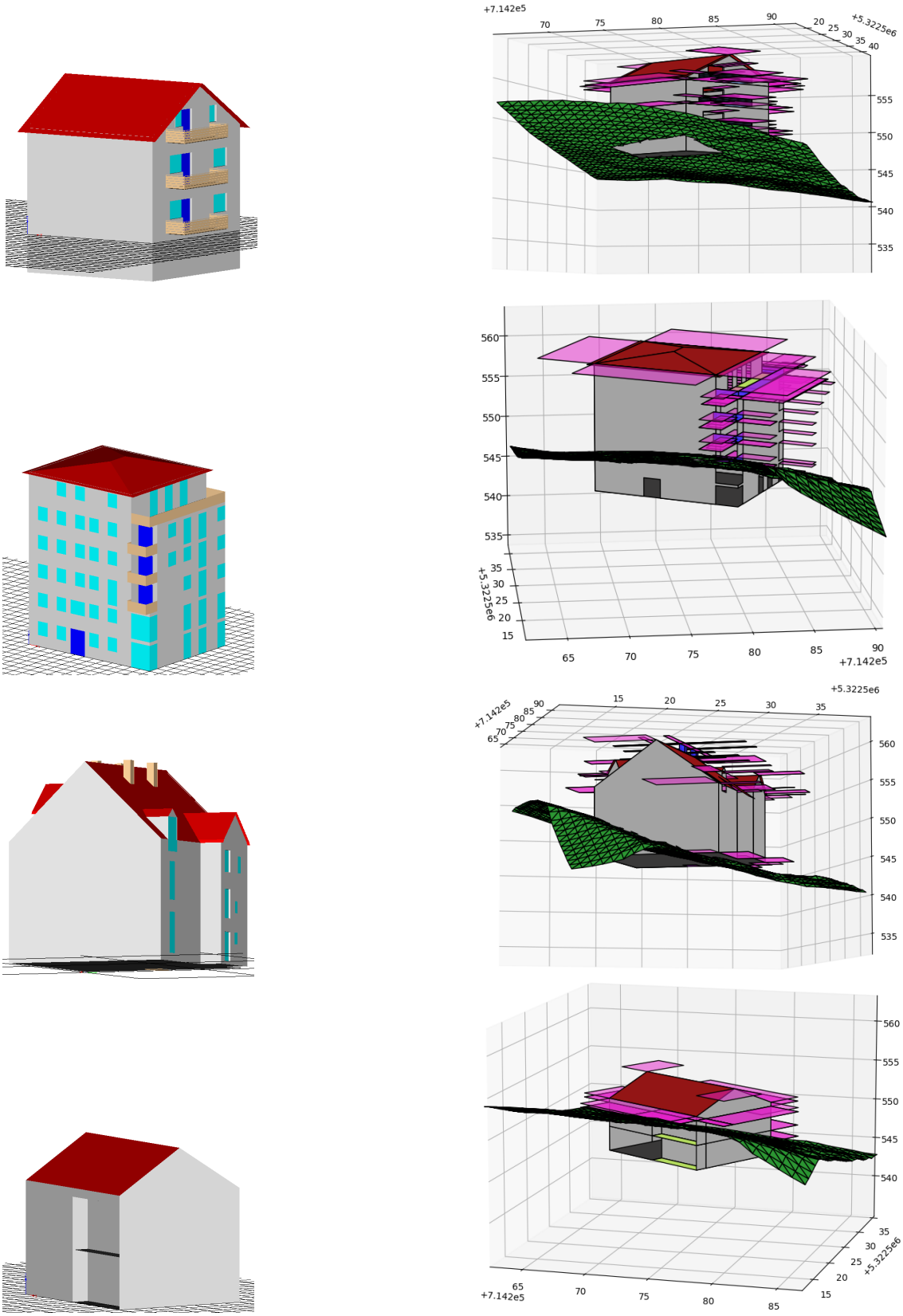


Abbildung 75 Anwendung des Algorithmus zur automatischen Berechnung von Abstandsflächen auf verschiedene Gebäudemodelle.

5 Erstellung eines 3D-Lageplans

5.1 Workflow

Mittels des in den vorigen Kapiteln entwickelten SiteplanADE-Datenmodells, sowie des Algorithmus zur Abstandsflächenberechnung, soll nun exemplarisch, von einem konventionellen CAD-Lageplan ausgehend, ein 3D-Lageplan im CityGML-Format erstellt werden. Das Vorgehen entspricht einer Konvertierung des zweidimensionalen CAD-Lageplans in einen semantischen Lageplan, der die Lageplan-Objekte mit 3D-Geometrie ergänzt.
Da Objekte in CAD-Dateien oft nicht konsistent thematische Objektgruppierungen aufweisen, wird der CAD-Lageplan zunächst in einer GIS-Software bearbeitet. Mittels eines DOPs werden zusätzliche Objektarten wie bspw. Landbedeckung, private Verkehrsflächen, etc., digitalisiert. Die Lageplan-Objekte werden anschließend je nach Objektart in sogenannte *Layer* eingeteilt und als Shape-Dateien exportiert.
Zusammen mit dem DGM, dem DOM* und dem Gebäudemodell werden die Shape-Dateien in eine FME eingelesen (die in dieser Arbeit verwendete FME stammt von der Firma Safe Software [`https://www.safe.com/`]). Zusätzliche Daten, welche für die Erstellung des 3D-Lageplans notwendig sind (z.B. LOD2-Gebäude der Nachbargrundstücke, sowie ALKIS-Flurstücke) werden von Online-Portalen der Landesvermessungsämter bezogen und ebenfalls in die FME hinzugeladen.
Für wiederkehrende Objekte, wie Bäume oder Straßenlaternen, werden außerdem 3D-Modelle verwendet. Diese erlauben die Anwendung des Konzepts der impliziten Geometrie. Dafür wird ein Referenzobjekt mit expliziter Geometrie instanziiert. Die restlichen Objekte enthalten nur einen Referenzpunkt im Bezugskoordinatensystem, eine Transformationsmatrix zur Anpassung der referenzierten Geometrie, sowie die Referenz zur Geometrie des instanziierten Objekts (Gröger et al., 2012).
In der FME werden die 2D-Lageplan-Objekte über das DGM bzw. das DOM* mit 3D-Geometrie ergänzt. Außerdem wird die Abstandsflächenberechnung anhand des geplanten Gebäudemodells durchgeführt und schließlich sämtliche Lageplan-Objekte auf das Schema der SiteplanADE abgebildet und in eine Datei im CityGML-Format geschrieben.
In den folgenden Abschnitten wird erläutert, wie die in Abbildung 76 gezeigten Arbeitsschritte realisiert werden.

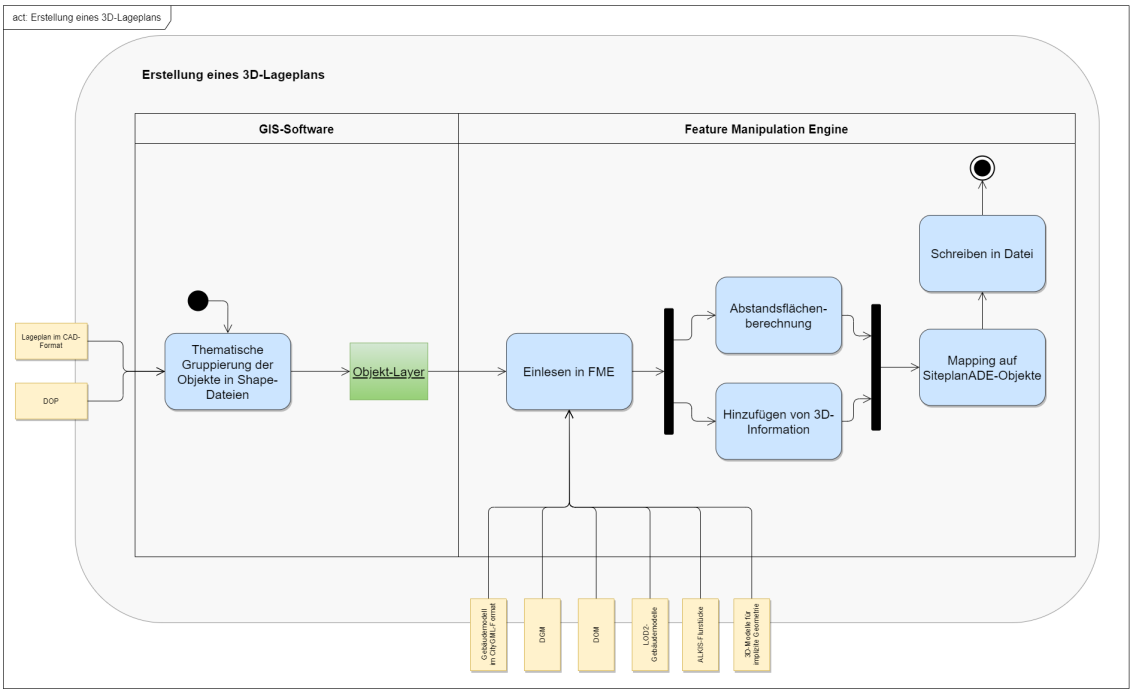


Abbildung 76 Workflow für die Erstellung des 3D-Lageplans.

5.2 Durchführung/Implementierung des Workflows

5.2.1 Vorverarbeitung des CAD-Lageplans

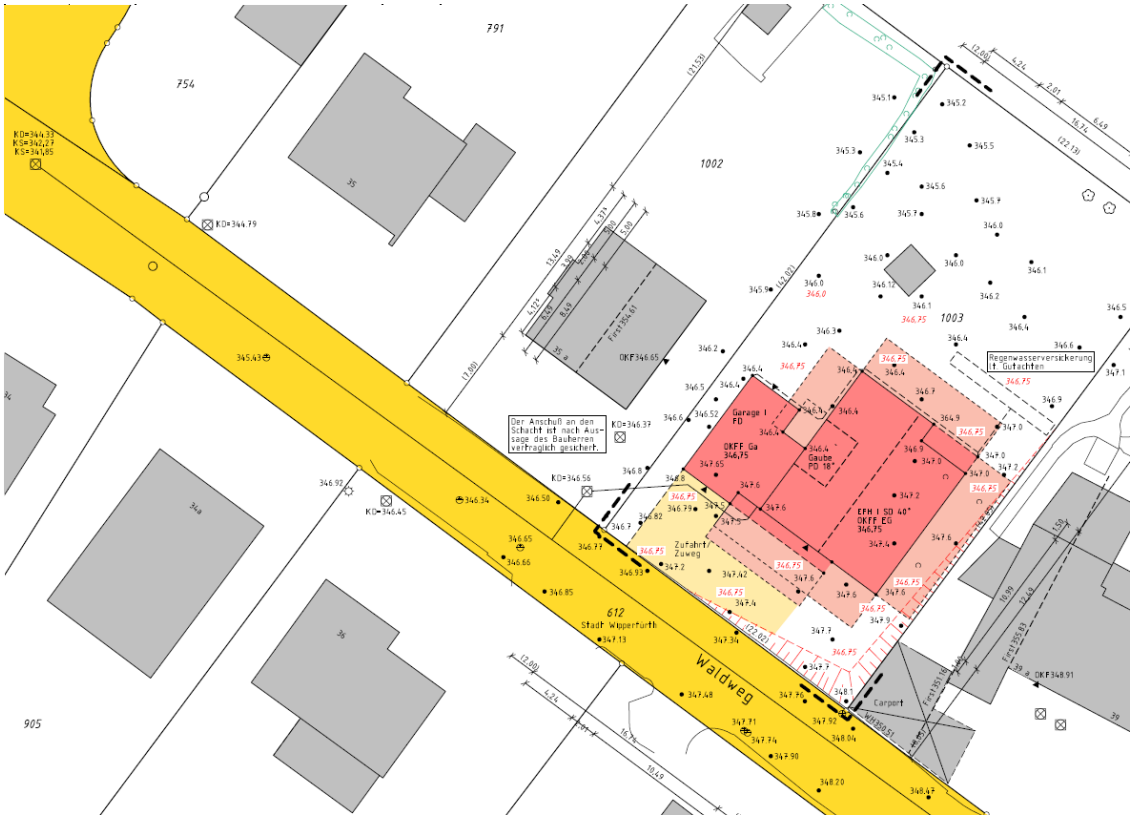


Abbildung 77 Ausschnitt aus dem vom BDVI zur Verfügung gestellten Lageplan.

Der Beispieldatensatz eines CAD-Lageplans wird vom BDVI bereitgestellt (siehe Abbildung 77). In diesem Lageplan sind folgende Objektarten enthalten:

- Kanaldeckel
- Straßenkappe
- Hausanschluss
- Abwasserkanal
- Regenwasserversickerung
- Straßenlaterne
- Flurstück
- Grenzpunkt
- Vermessungspunkt
- Verkehrsfläche (existierend/geplant/öffentlich/privat)
- Individuelles Vegetationsobjekt
- Flächige pflanzliche Bedeckung
- Nachbargebäude
- Geplante bauliche Anlage
- (Landbedeckung)

In diesem konkreten Fall befindet sich das geplante Objekt außerhalb eines durch den Bebauungsplan festgelegten Baugebiets, sodass keine Objekte aus dem Bebauungsplan vorhanden sind.

Die Objektarten liegen in der CAD-Datei nicht gruppiert vor. Es ist daher notwendig, die geometrischen Objekte der CAD-Zeichnung thematischen Objektarten zuzuordnen, bevor die eigentliche Konvertierung in einen 3D-Lageplan stattfinden kann.
Für diesen Zweck wird der CAD-Lageplan in die GIS-Software QGIS geladen. Dort werden die geometrischen Objekte verschiedenen Objektarten-Layern zugewiesen. Bei der Gruppierung der Objekte in Layer wird bereits darauf geachtet, dass Objekte im selben Layer dieselbe Zielklasse im CityGML-Standard aufweisen.

Objektart	Anwendungsschema	Zielklasse
Kanaldeckel	UtilityNetworkADE	TerminalComponent
Straßenkappen	UtilityNetworkADE	TerminalComponent
Hausanschluss	UtilityNetworkADE	TerminalComponent
Straßenkappen	UtilityNetworkADE	TerminalComponent
Regenversickerungsfläche	UtilityNetworkADE	OtherComponent
Abwasserkanal	UtilityNetworkADE	Canal
Straßenlaterne	CityGML::CityFurniture	CityFurniture
Flurstück	SiteplanADE	AX_Flurstueck
Grenzpunkt	SiteplanADE	AX_Grenzpunkt
Vermessungspunkt	SiteplanADE	AX_Aufnahmepunkt
Verkehrsfläche (existierend/geplant/öffentlich/privat)	CityGML::Transportation	TrafficArea
Einzelne Vegetationsobjekte	CityGML::Vegetation	SolitaryVegetationObject
Flächige pflanzliche Bedeckung	CityGML::Vegetation	PlantCover
Nachbargebäude	CityGML::Building	Building
Geplante bauliche Anlage	CityGML::Building	Building
Landbedeckung	CityGML::LandUse	LandUse

Tabelle 13 CityGML-Zielklassen der im Lageplan enthaltenen Objektarten.

Jedes Objekt erhält ein Attribut ”height_source”, dass widerspiegelt welche Datenquelle später verwendet werden soll, um die 3D-Geometrie zu erlangen. Das hat den Hintergrund, dass das vom BDVI bereitgestellte DOM* bspw. für die Gewinnung der Höheninformation für Vegetationsobjekte, wie z.B. Hecken, verwendet werden wird. Straßenflächen jedoch würden ihre Höheninformation vom DGM beziehen. Vermessungspunkte des CAD-Lageplans wurden mit Text versehen, der die jeweilige Höhe des Punktes repräsentiert. Damit sind drei Datenquellen für 3D-Information vorhanden, sodass jedes Objekt spezifizieren muss, auf welche Datenquelle es sich in der FME beziehen wird. Je nach Objekt-Geometrie wird auch noch unterschieden, ob es sich bei der zu extrahierenden 3D-Geometrie um eine Punkt - oder TIN-Geometrie handelt.

Das Attribut ”height_source” nimmt folglich einen der Werte

- ”tin_dtm” für die flächenhafte Zuweisung des korrespondierenden Ausschnitts aus dem DGM als TIN,
- ”point_cloud_dtm” für die flächenhafte Zuweisung des korrespondierenden Ausschnitts aus dem Punktwolken-DOM* als TIN,
- ”point_height_dtm” für die punktweise Extraktion der Höhe aus dem DGM,
- ”point_height_point_cloud” für die punktweise Extraktion der Höhe aus dem Punktwolken-DOM*,
- ”heighttext” für die Extraktion der Höheninformation aus einem Text-Attribut des Objekts

an.
Die Flurstücksgrenzen werden im CAD-Datensatz nur ausschnittsweise und unvollständig dargestellt. Darum werden diese im aktuellen Arbeitsschritt nicht berücksichtigt, sondern vom Landesvermessungsamt als ALKIS-Datensatz bezogen und später direkt in die FME geladen. Neben den Flurstücken werden auch die Nachbargebäude in diesem Schritt vernachlässigt. Da von den Landevermessungsämtern flächendeckend LOD2-Gebäude zur Verfügung stehen, werden auch diese von einem Online-Portal bezogen und erst später in der FME verarbeitet.
Unter Hinzuladen eines DOPs werden Landbedeckungsflächen digitalisiert, mit einem die Art der Landbedeckung spezifizierenden Attribut versehen, und in dem entsprechenden Layer abgespeichert.

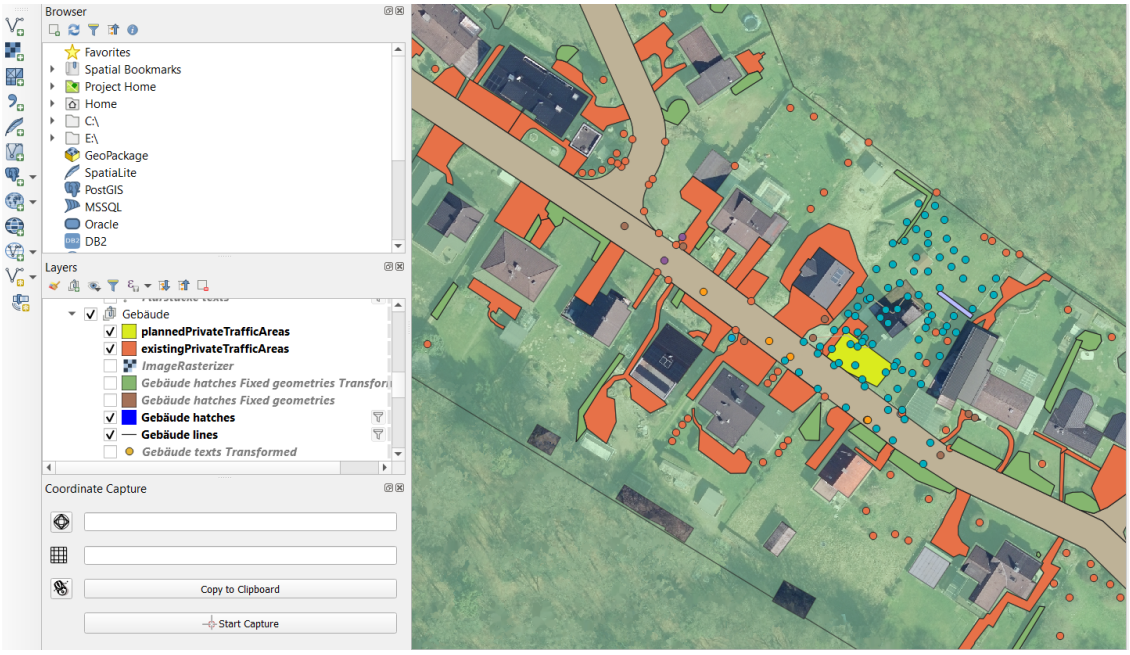


Abbildung 78 Unter zu Hilfenahme eines DOP werden zusätzlich die nicht im Lageplan enthaltenen Landbedeckungsflächen digitalisiert. Flächen des geplanten Gebäudes, sowie der Gebäude-Footprints werden mittels entsprechender (ALKIS-)Shape-Dateien ausgespart.

Nach dem Export aus QGIS liegen 16 Shape-Dateien vor:

Dateiname	Geometrie der Shape-Datei	Beschreibung/Inhalt
border_points.shp	Punkt	Grenzpunkte aus dem CAD-Lagelan.
canals.shp	Linie	Abwasserkanäle aus dem CAD-Lagelan.
control_points.shp	Punkt	Vermessungspunkte aus dem CAD-Lageplan.
drain_covers.shp	Punkt	Kanaldeckel aus dem CAD-Lageplan.
existingPrivateTrafficAreas.shp	Polygon	Vorhandene private Verkehrsflächen. Teils aus dem CAD-Lageplan, z.T. (zum Teil) digitalisiert mit Hilfe des DOPs.
existingPublicTrafficAreas.shp	Polygon	Vorhandene öffentliche Verkehrsflächen. Teils aus dem CAD-Lageplan, z.T. digitalisiert mit Hilfe des DOPs.
height_control_points.shp	Punkt	Höhenpunkte aus dem CAD-Lageplan.
land_use.shp	Polygon	Landbedeckung. Digitalisiert mittels des DOPs.
planned_house_connection.shp	Punkt	Geplante Hausanschlüsse aus dem CAD-Lageplan.
plannedPrivateTrafficAreas.shp	Polygon	Geplante private Verkehrsflächen aus dem CAD-Lageplan.
plant_cover.shp	Polygon	Flächige Vegetation (z.B. Hecken). Teils aus dem CAD-Lageplan, teils digitalisiert mit Hilfe des DOPs.
propertyParcel.shp	Polygon	Grundstücksgrenzen aus dem CAD-Lageplan.
rain_water_sink.shp	Polygon	Fläche für die Regenwasserversickerung aus dem CAD-Lageplan.
street_caps.shp	Punkt	Kanaldeckel aus dem CAD-Lageplan.
street_light.shp	Punkt	Straßenlaterne aus dem CAD-Lageplan.
vegetation_trees.shp	Punkt	Baumpositionen aus dem CAD-Lageplan.
siteplan_bounds.shp	Polygon	Begrenzungsrahmen des Lageplans.

Tabelle 14 Resultierende Shape-Dateien nach der Verarbeitung des CAD-Lageplans in QGIS.

5.2.2 Prozessierung in der Feature Manipulation Engine

Einlesen der Daten

Nachdem die Shape-Dateien, für die verschiedenen Objektarten des Lageplans erzeugt wurden, können alle notwendigen Daten zusammengetragen werden, um diese in eine FME einzulesen:

Datensatz	Datenformat	Datenherkunft/-erzeugung
Dateien der Lageplan-Objektarten	Shape	CAD-Zeichnung des Lageplans
Gebäudemodell des geplanten Bauwerks	CityGML	CAD-Zeichnung [1]
LOD2-Gebäudemodelle der Nachbargebäude	CityGML/ALKIS	Online-Portal des Landesvermessungsamts
DOM* des Gebiets des Lageplans	LAS-Punktwolke	BDVI/Laserscanning
DGM des Gebiets des Lageplans	GeoTIFF	Online-Portal des Landesvermessungsamts
Flurstücke im Gebiet des Lageplans	ALKIS	Online-Portal des Landesvermessungsamts

Tabelle 15 Eingangsdaten mit Format und Herkunft.

Für das Einlesen der unterschiedlichen Datenformate müssen in FME die entsprechenden *Reader* verwendet werden. Nachdem die Datensätze eingelesen wurden, stehen sie als sogenannte *Features* in einer Datenformat-neutralen internen Darstellung innerhalb der FME für die weitere Verarbeitung und Transformation bereit.

[1] Siehe Anhang E für eine Anleitung zur Konvertierung eines CAD-Gebäudeentwurfs in das CityGML-Format mittels des SketchUp Plugins *GEORES*.

Abstandsflächenberechnung

Für den Quellcode der Implementierung des Algorithmus der Abstandsflächenberechnung wird auf Unterabschnitt 4.2.8 verwiesen.
In Abbildung 79 sind die (*Feature*-)Datenströme zu sehen, die in den *PythonCaller*-Transformer fließen. Da der Objektfluss in den *PythonCaller* seriell geschieht und es nicht möglich ist, verschiedene *Input-Ports* für verschiedene *Features* zu definieren, müssen den Eingangsobjekten über den *AttributeCreator* identifizierende Attribute hinzugefügt werden, um diese zu unterscheiden. Für die Identifizierung eines *Features* wird das Attribut mit dem Namen ”cscObject” verwendet. In diesem Attribut ist der Schlüsselname des Eingangsobjekts hinterlegt. Im Falle des Grundstücks (*property_parcel*) ist der Wert dieses Attributs ”propertyParcel”, für DGM-Punkte nimmt das Attribut den Wert ”dtmPoint” an. In der *input*-Funktion des *Pyhthon-Callers* werden die eingehenden *Features* dann entsprechend ihrem Identifikator-Attribut weiterverarbeitet.
Bevor die Punkte des DGMs jedoch in den *PythonCaller* eingehen, wird das ursprüngliche DGM mittels des *Clipper*-Transformers auf die Punkte innerhalb der Grenzen des Lageplans (*siteplan_bounds*) begrenzt.
Das Gebäudemodell für die Abstandsberechnung geht nicht als FME-*Feature* in den *PythonCaller* ein, sondern wird direkt aus dem Dateisystem gelesen, obwohl das geplante Gebäudemodell auch in die FME-*Workbench* geladen wird. Das hat den Hintergrund, dass der Abstandsflächenalgorithmus das Gebäudemodell in der in Abschnitt 4.2.3 beschriebenen Repräsentation benötigt.
Damit das Gebäudemodell jedoch mit dem übereinstimmt, welches in die FME-*Workbench* eingelesen wird, wird der Dateipfad des Gebäudemodells aus dem globalen *fme.macroValues-Dictionary*-Objekt entnommen.

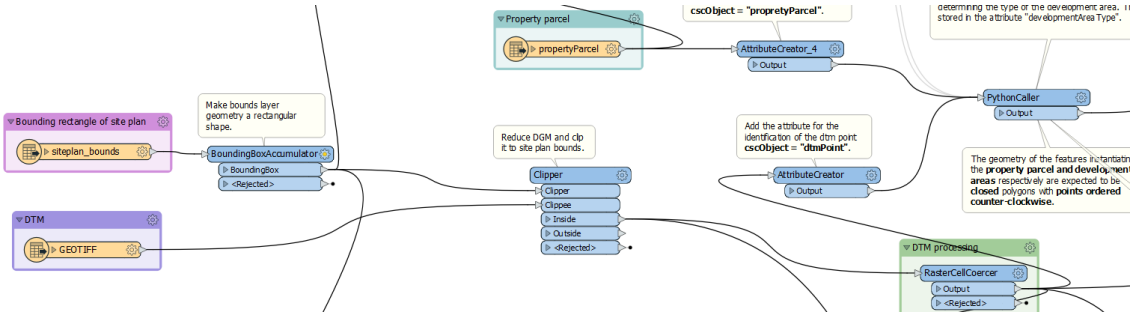


Abbildung 79 Ausschnitt aus der FME-Workbench. Zu sehen sind die Eingangsdatenströme in den *PythonCaller*, in welchem die Abstandsflächenberechnung durchgeführt wird.

Der *PythonCaller* gibt in seiner *output*-Funktion, nachdem die Berechnungen bzgl. der Abstandsflächen abgeschlossen sind, unterschiedliche *Features* aus. Zum einen *Features*, die die Geometrie der berechneten Abstandsflächen beinhalten, zum anderen die Gebäudeschnittlinien des Gebäudemodells mit dem Gelände, welche im Zuge der Abstandsflächenberechnung ebenfalls berechnet wurden.

Alle Ausgangs-Objekte referenzieren über eine Eltern-ID (Identifikator) (*gml_parent_id*) die zugehörige Gebäudekomponente, in der sie enthalten sind bzw. zu der sie gehören. Für die einzelnen Abstandsflächen bedeutet das die GML-ID (*gml_id*) der thematischen Begrenzungsflächen, für die Gesamtabstandsfläche, sowie für die Geländeschnittlinien die GML-ID (*gml_id*) des *Building*-Elements.
Um die ausgegebenen *Features* filtern zu können, wird diesen ein Attribut namens ”clearanceSpaceType” hinzugefügt. Dieses nimmt die Werte

- ”hull” für die Gesamtabstandsfläche des Gebäudes, oder
- ”buildingTIC” für die Geländeschnittlinien

an. Die Abstandsflächen-*Features* der einzelnen Gebäudekomponenten erhalten dieses Attribut nicht. Sie werden darüber identifiziert, dass ihnen dieses Attribut fehlt (*<Missing>*).
Die Gesamtabstandsfläche des Gebäudes wird über den *Extruder*-Transformer in eine Solid-Geometrie umgewandelt, welche dann den Abstandsraum des Gebäudes repräsentiert.
Sowohl für die einzelnen Abstandsflächen, als auch den Abstandsraum wird das Koordinatenreferenzsystem festgelegt. Außerdem wird die Geometrie-Eigenschaft (*property*) der *Features* entfernt und im GML3.2-Format in einem Text-Attribut gespeichert.

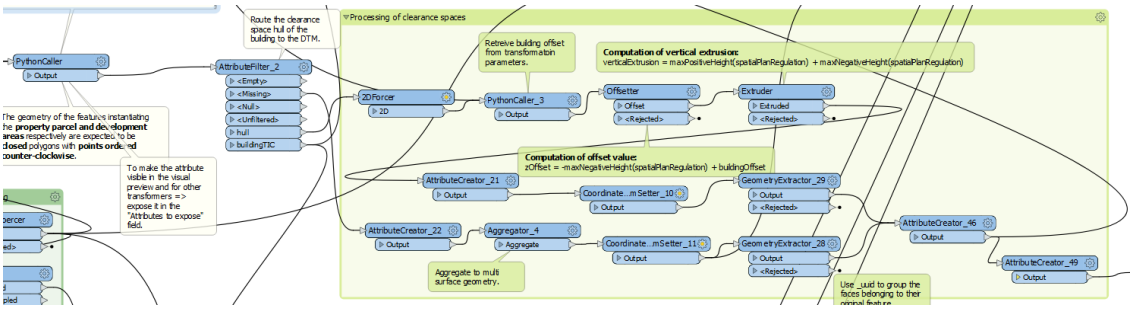


Abbildung 80 Ausschnitt aus der FME-Workbench. Die aus dem *PythonCaller* resultierenden *Features* werden basierend auf dem ”clearanceSpaceType” gefiltert und weiterverarbeitet.

Die Abstandsflächen-*Features* werden nun mit den Gebäudekomponenten des Gebäudemodells re-assoziiert. Das geschieht mit Hilfe des *FeatureMerger*-Transformers. Dieser wendet das *join on*-Konzept aus dem Bereich der Datenbanken an. Basierend auf der Eltern-ID (*gml_parent_id*) der Abstandsflächen-*Features* und der GML-ID (*gml_id*) werden die Attribute der zusammenpassenden *Features* fusioniert. Die *Requestor-Features* sind dabei die Gebäudekomponenten, in den *Supplier-Port* gehen die Abstandsflächen-*Features* ein.

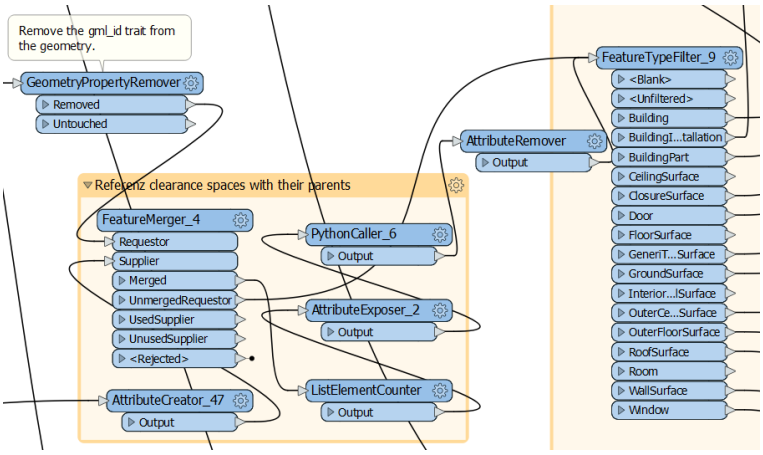


Abbildung 81 Über den Transformer *FeatureMerger* werden die Abstandsflächen über die Attribute ”parent_id” mit den zugehörigen Gebäudekomponenten und deren Attribut ”gml_id” referenziert.

Hinzufügen von 3D-Information

Wie in Unterabschnitt 5.2.1 beschrieben, enthält jedes Objekt ein Attribut, welches spezifiziert, aus welcher Datenquelle es die 3D-Information bezieht, um seine 3D-Geometrie zu konstruieren. Die eingehenden Daten werden anhand dieses Attributs gefiltert und weiterverarbeitet.

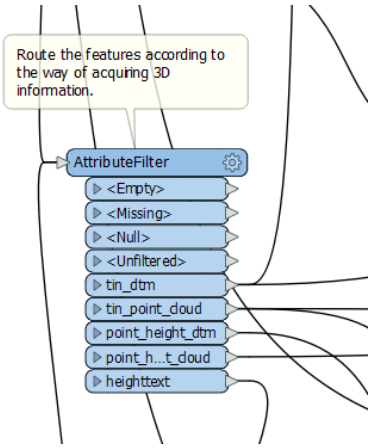


Abbildung 82 Filterung der *Features* basierend auf dem Attribut zur Gewinnung von 3D-Information.

Objekte mit Punktgeometrie

Höhe aus dem DGM: Um die Punkthöhe aus dem DGM zu extrahieren wird auf den *PointOnRasterValueExtractor*-Transformer zurückgegriffen. Dieser interpoliert die Grauwerte des DGMs für die gegebene Punktposition und speichert sie einem Attribut. Dieses Attribut wird später im *3DForcer*-Transformer verwendet, um der z-Koordinate einen Wert zuzuweisen.

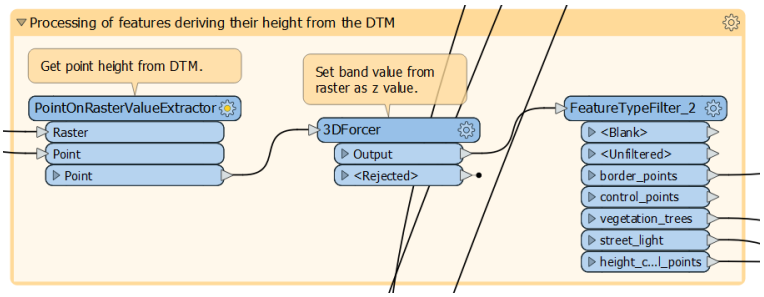


Abbildung 83 Darstellung des Unterschieds zwischen DGM (graue Linie) und DOM* (blaue Linie) (Waser, 2020).

Betroffene Lageplan-Objektart:

- Grenzpunkt
- Kanaldeckel
- Straßenkappe

*Höhe aus dem Punktwolken-DOM**: Das Punktwolken-DOM* wird nur für die Höhenextraktion von Objekten verwendet, welche nicht im DGM enthalten, sind (z.B. für Vegetation). Da in diesem Lageplan nur Punktobjekte vorhanden sind, welche sich auf der Erdoberfläche befinden, kommt es zu keiner punktweisen Extraktion von Höhen aus dem Punktwolken-DOM*.

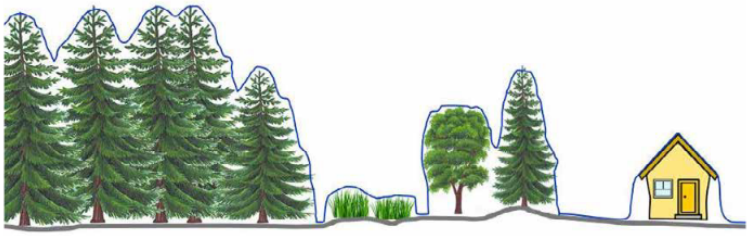


Abbildung 84 Darstellung des Unterschieds zwischen DGM (graue Linie) und DOM* (blaue Linie) (Waser, 2020).

Höhe aus dem Text-Attribut: Für Objekte mit der Datenquelle aus ihrem eigenen Text-Attribut wird der Transformer *3DForcer* genutzt. Dieser ergänzt die Geometrie um die dritte Komponente und weist dieser den Wert der als Text bereitgestellten Höhe zu.

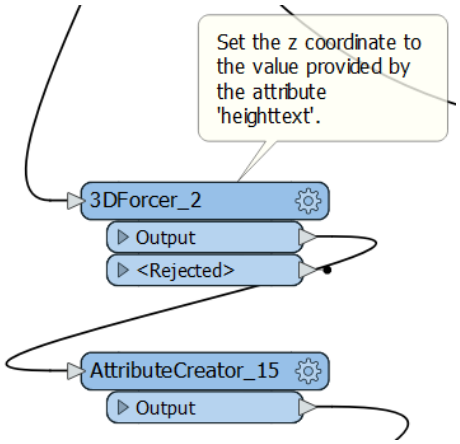


Abbildung 85 Die *Feature*-Geometrie wird um die z-Komponente ergänzt. Der Wert wird aus dem Text-Attribut extrahiert, welches den Höhenwert beinhaltet.

Betroffene Lageplan-Objektart:

- Vermessungspunkt: Lage-/Höhenpunkte

Objekte mit impliziter Geometrie

Objekte mit impliziter Geometrie beziehen ihre Höheninformation zunächst aus einer der vorher beschrieben Höheninformationsquellen (DGM, Punktwolken-DOM* oder Text-Attribut). Danach werden die Punktkoordinaten in Text-Attributen gespeichert (*CoordinateExtractor*) und die Geometrie des *Features* entfernt (*GeometryRemover*). Anschließend wird den *Features* ein Schlüssel-Attribut hinzugefügt, um diese im Folgenden mittels des *FeatureMergers* mit dem entsprechenden 3D-Modell der impliziten Geometrie zu fusionieren. Dabei übernehmen die *Features* neben den Attributen auch die Geometrie der 3D-Modelle.

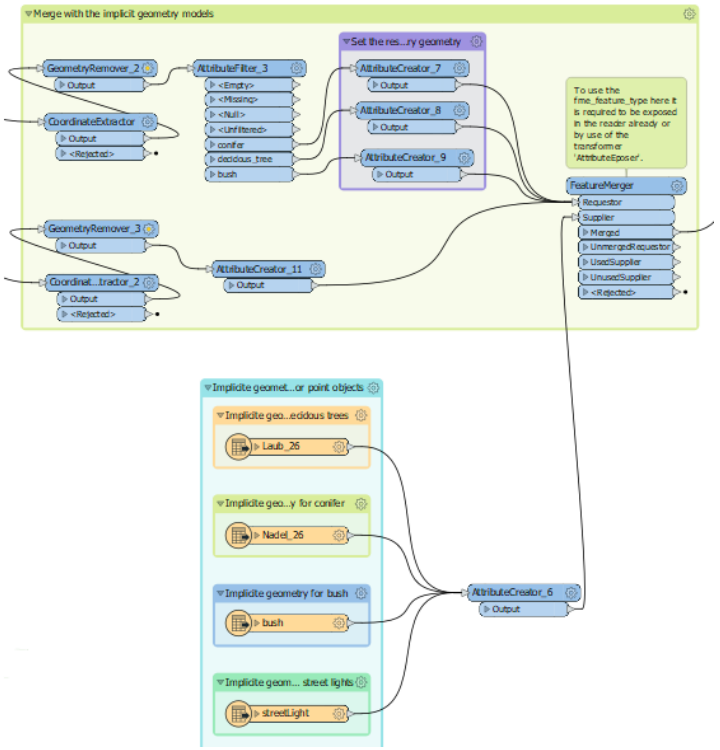


Abbildung 86 Referenzierung der *Features* mit den korrespondieren 3D-Bibliothek-Modellen.

Im nächsten Schritt wird mittels des *AttributeCreator* ein Attribut, welches die Transformationsmatrix enthält, erstellt. Diese ist eine 4x4-Projektionsmatrix, welche auf homogene Koordinaten $[X, Y, Z, W]^T$ angewendet wird:

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

Die Matrix-Elemente *A, F, K* sind für die Skalierung zuständig. Die Translationselemente *D, H, L* nehmen die Koordinaten des ursprünglichen Punktorts an. *N* hat den Wert 1, die restlichen Matrix-Elemente sind in diesem Fall 0-Einträge (keine Rotation), sodass sich folgende Matrix ergibt:

$$T = \begin{bmatrix} S_x & 0 & 0 & \_X \\ 0 & S_y & 0 & \_Y \\ 0 & 0 & S_z & \_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Anschließend wird die erhaltene Geometrie des 3D-Modells von dem *Feature* entfernt und in Textform im GML3.2-Format in einem Attribut gespeichert. Danach wird ein Referenzpunkt erstellt (*VertexCreator*). Dieser liegt im Ursprung, da die Punktkoordinaten für die Translation bereits in der Transformationsmatrix enthalten sind. Auch die Geometrie des Referenzpunkts wird im Anschluss de-referenziert und als XML-Fragment im GML3.2-Format abgespeichert. Über den *Counter*-Transformer werden die *Features* mit Indizes versehen. So ist es möglich, das *Feature* mit Index 0 als Referenzobjekt zu verwenden. Die GML-ID (*gml_id*) des Referenzobjektes wird dann in die verbleibenden *Features* als Attribut für die Geometrie-Referenzierung eingefügt (*FeatureMerger*).

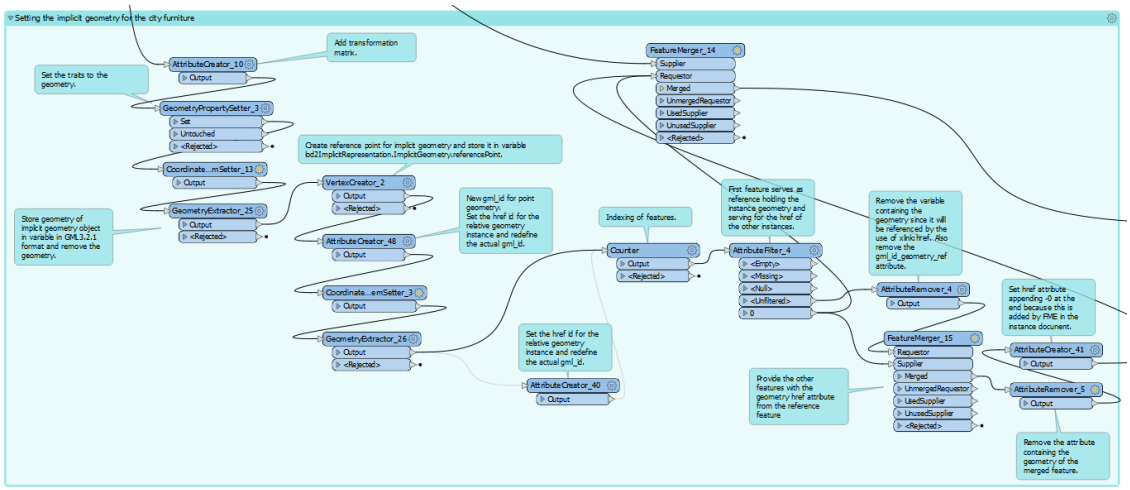


Abbildung 87 Referenzierung der *Features* werden mit der Instanz-Geometrie.

Betroffene Lageplan-Objektart:

- Individuelles Vegetationsobjekt: Bäume (Nadel/Laub), Büsche
- Straßenlaterne

Objekte mit Liniengeometrie

Höhe aus dem DGM: In die Liniensegmente werden in Intervallen von 5 m Linienpunkte eingefügt. Die zusätzlichen Punkte bewirken, das durch die kleinere Schrittweite, die Linien dem Geländeverlauf besser folgen werden.
Im nächsten Schritt passt der *SurfaceDraper*-Transformer nun die z-Komponenten der Linienpunkte derart an, dass die Linien die Form des Geländeverlaufs (DGM) annehmen.

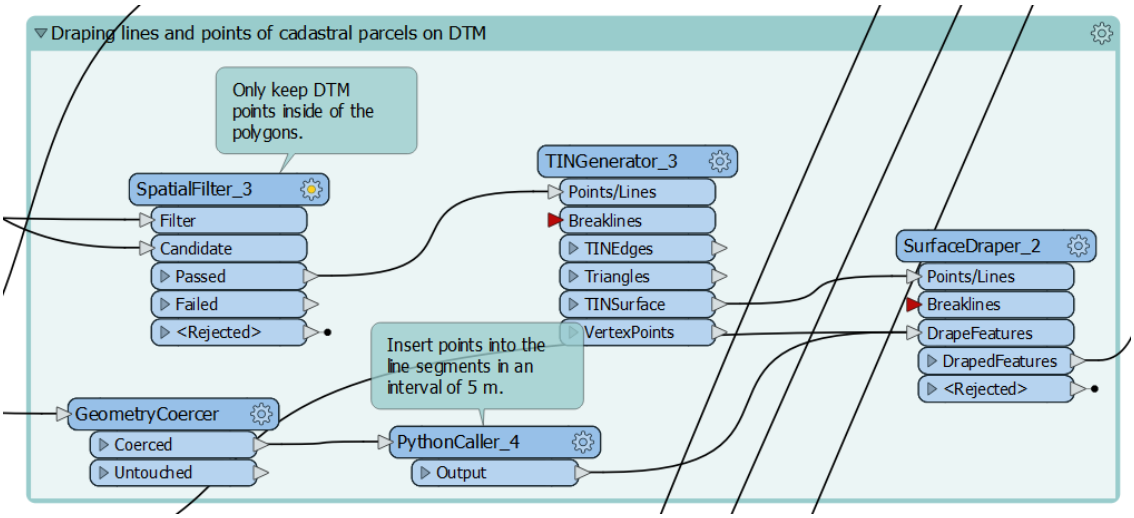


Abbildung 88 Drapieren der Liniensegmente über die TIN-Oberfläche.

Betroffene Lageplan-Objektart:

- Flurstücksgrenze

*Höhe aus dem Punktwolken-DOM**: Da im Beispieldatensatz des Lageplans keine Linienobjekte vorhanden sind, die sich auf Objekte über der Erdoberfläche beziehen, findet keine Extraktion von Höhen aus dem Punktwolken-DOM* statt.

Höhe aus dem Text-Attribut: Die Prozessierung der linienhaften Objekte mit Höheninformation aus Text-Attributen ist identisch zu der, der Punktgeometrie-Objekte. Über den Transformer *3DForcer* wird die Liniengeometrie auf drei Dimensionen erweitert und der Wert für die z-Komponente aus dem entsprechenden Text-Attribut übernommen.

Betroffene Lageplan-Objektart:

- Abwasserkanal

Objekte mit Flächengeometrie

Oberfläche aus dem DGM: Für die flächenhafte Extraktion aus dem DGM werden die DGM-Punkte anhand der Flächengeometrie der Filter-*Features* gefiltert (*SpatialFilter*). Nur die Punkte innerhalb der Flächengeometrie der *Features* werden für die Weiterverarbeitung beibehalten. Mittels des *TINGenerators* wird aus den verbliebenen Punkten eine Oberfläche modelliert. Diese wird mit Hilfe der Filter-*Features* derart zugeschnitten (*Clipper*), dass für jedes Eingangs-*Feature* ein TIN korrespondierend zu dessen Flächengeometrie entsteht. Bei der Schneide-Aktion wird jeder zugeschnittene Geometrie die GML-ID (*gml_id*), sowie sämtliche anderen Attribute des für die Schneide-Operation verwendeten *Features* hinterlegt.

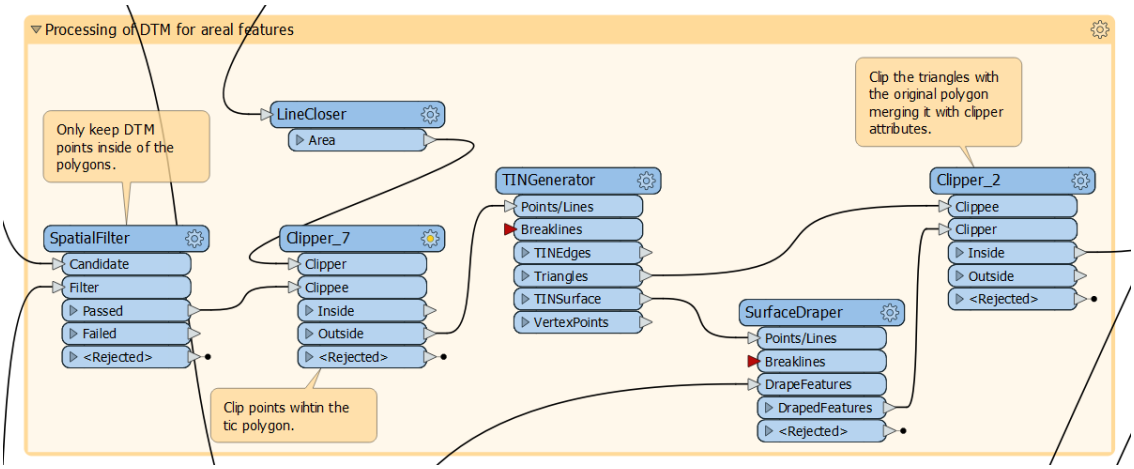


Abbildung 89 Filterung der DGM-Punkte und Zuschneiden der entstehenden TIN-Oberfläche.

Nun werden die Oberflächen in ihre einzelne Komponenten zerteilt (*Deaggregator*). Als nächstes dient die hinterlegte GML-ID (*gml_id*) der erneuten Zusammenfügung der Oberflächenbestandteile gemäß des für die Schneide-Operation verwendeten *Features* (*Aggregator*).

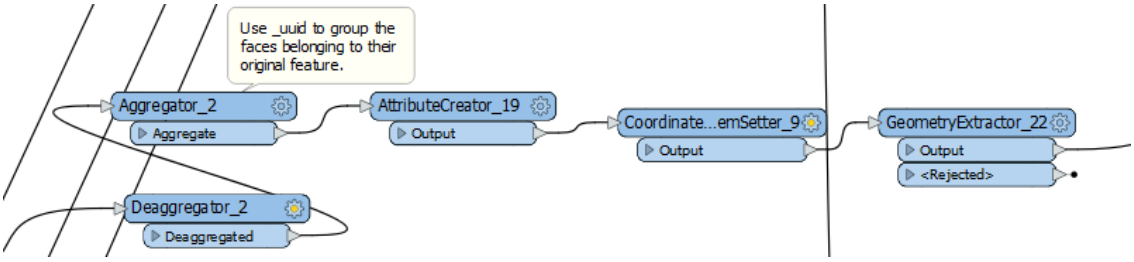


Abbildung 90 Zusammenfügung der zusammengehörenden Oberflächenbestandteile gemäß des für die Schneide-Operation verwendeten *Features*.

Betroffene Lageplan-Objektart:

- Landbedeckung: Grass, Wald, Wasser
- Verkehrsfläche: öffentliche Straßen, private Flächen (Einfahrten, Wege, ...)

Oberfläche aus dem Punktwolken-DOM:* Die Extraktion von Oberflächen aus dem Punktwolken-DOM*s ist ähnlich zu der Prozessierung einer Oberfläche aus dem DGM. Nach der Vereinfachung der Punktwolke und anschließender TIN-Generierung wird die TIN-Oberfläche mittels der Eingangs-*Features* zugeschnitten. Es resultieren wieder je verwendeten *Features* eine Oberfläche, welche alle Attribute des schneidenden *Features* enthält. Auch hier werden die Oberflächen zerteilt und wieder zusammengefügt, wobei die Oberflächenbestandteile ebenfalls gemäß der GML-ID (*gml_id*) zusammengefasst werden.

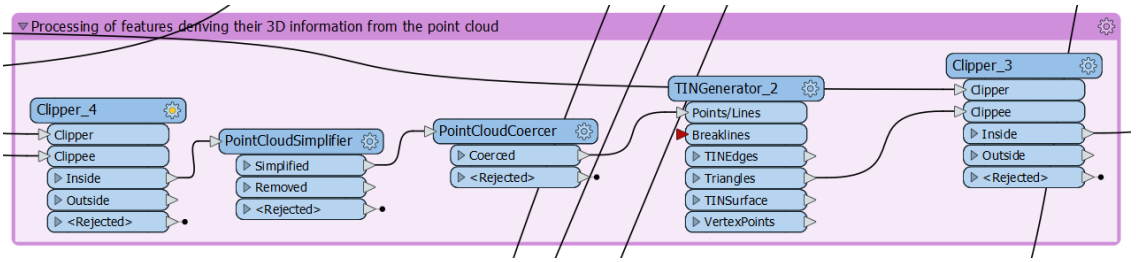


Abbildung 91 Extraktion von Oberflächen aus dem Punktwolken-DOM*.

Betroffene Lageplan-Objektart:

- Flächige pflanzliche Bedeckung: Hecken
- Regenversickerungsfläche

Mapping auf SiteplanADE-Schema und Schreiben der Ausgabe-Datei

Um schlussendlich die bearbeiteten *Features* in eine Datei im Ziel-Datenformat schreiben zu können, muss in der FME der entsprechende *Writer* verwendet werden. Das Zielformat ist CityGML, welches ein GML-Anwendungsschema darstellt. Deshalb wird der GML-*Writer* genutzt. In diesem muss der Pfad zum verwendeten Anwendungsschema gesetzt werden. Für das Schreiben des Lageplans wird folgendes CityGML-Profil verwendet:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.opengis.net/citygml/profiles/base/3.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/citygml/profiles/base/3.0" elementFormDefault="qualified" attributeFormDefault="
      unqualified" version="3.0.0">
  <xs:import namespace="http://www.opengis.net/citygml/appearance/3.0" schemaLocation="./appearance.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/bridge/3.0" schemaLocation="./bridge.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/building/3.0" schemaLocation="./building.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityfurniture/3.0" schemaLocation="./cityFurniture.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/cityobjectgroup/3.0" schemaLocation="./cityObjectGroup.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/construction/3.0" schemaLocation="./construction.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/pointcloud/3.0" schemaLocation="./pointCloud.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/3.0" schemaLocation="./cityGMLBase.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/dynamizer/3.0" schemaLocation="./dynamizer.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/generics/3.0" schemaLocation="./generics.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/landuse/3.0" schemaLocation="./landUse.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/relief/3.0" schemaLocation="./relief.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/transportation/3.0" schemaLocation="./transportation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/tunnel/3.0" schemaLocation="./tunnel.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/vegetation/3.0" schemaLocation="./vegetation.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/versioning/3.0" schemaLocation="./versioning.xsd"/>
  <xs:import namespace="http://www.opengis.net/citygml/waterbody/3.0" schemaLocation="./waterBody.xsd"/>
  <!-- ADE schema files -->
  <xs:import namespace="http://www.citygml.org/ade/siteplan" schemaLocation="./SiteplanADE.xsd"/>
  <xs:import namespace="http://www.citygml.org/ade/utility/0.9.9" schemaLocation="./CityGML3.0_UtilityNetworkADE.xsd"/>
</xs:schema>
```

Listing 5.1: cityGMLProfile.xsd

Der *Writer* erzeugt nun eine interne Datenformat-neutrale Darstellung der Objekte des GML-Schemas bzw. des CityGML-Profils. Auf diese Objekt-Vorlagen (*Feature Types*) müssen die in der FME bearbeiteten *Features* abgebildet werden, damit sie in dem entsprechenden Format in die Ausgabedatei geschrieben werden können. Die Abbildung erfordert Kompatibilität sowohl bzgl. der Namen der Attribute, als auch des Datentyps der Attribute. Um dies zu bewerkstelligen, kann der *AttributeRenamer*-Transformer verwendet werden.

Dieser ermöglicht es, die Bezeichner von Attributen zu ändern, um dem des Ziel-Objektes zu entsprechen. Ebenso ist der *AttributeCreator* von Nutzen, wenn Attribute hinzugefügt werden sollen, die für die Validität des Zielformats erforderlich sind.
Die entsprechenden Namen der Attribute und deren Datentyp der Zielobjekte sind in den jeweiligen *Writer-FeatureTypes* unter den Reitern *User Attributes* und *Format Attributes* zu finden.

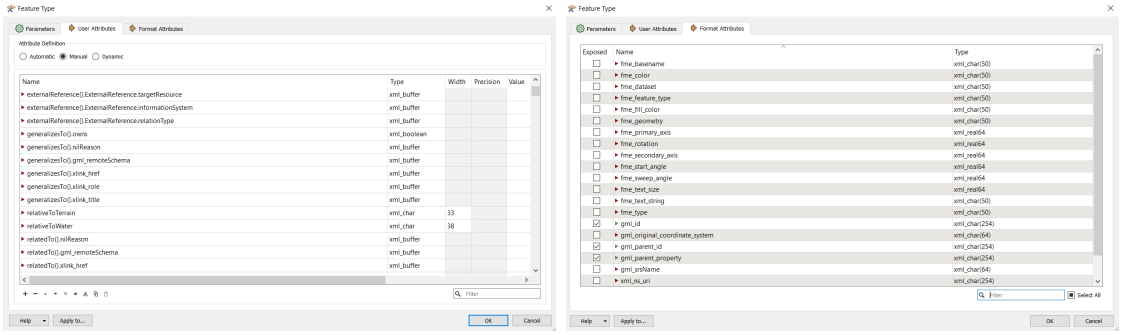


Abbildung 92 *Feature Type*-Reiter: *User Attributes* und *Format Attributes*.

Besondere Bedeutung kommt den Attributen ”gml_id”, ”gml_parent_id” zu.
Über die Attribute ”gml_id” und ”gml_parent_id” können dem GML-*Writer* die hierarchischen Beziehungen von Objekten mitgeteilt werden. Das Kind-Element referenziert im Attribut ”gml_-parent_id” die GML-ID des Eltern-Elements und stellt so die Beziehung her. In Spezialfällen muss der Name des Attributs des Eltern-Elements, in welchem das Kind-Element enthalten ist über das Attribut ”gml_parent_property” bereitgestellt werden.

Die Abbildung der FME-internen *Features* auf die Objekt-Templates (*Feature Types*) des GML-*Writers* erfolgt auf Basis der in Tabelle 13 dargestellten Abbildungsvorschriften von Objektart zu CityGML-Klasse.

Durch Ausführen der FME-*Workbench* startet die Prozessierung der Eingangsdaten und eine Datei gemäß dem SiteplanADE-Modell im CityGML-Format wird erzeugt.

CityGML-Ausgangsdatenmodell

Wohl in kaum einem Fall werden alle Pakete und Klassen, die von dem CityGML-Profil in Listing 5.1 für die Erstellung eines Lageplans eingebunden werden, benötigt. In Abbildung 93 ist das CityGML-Profil des Ausgangsdatensatzes abgebildet, welches für die Erstellung des Lageplans in dieser Arbeit benötigt wurde.

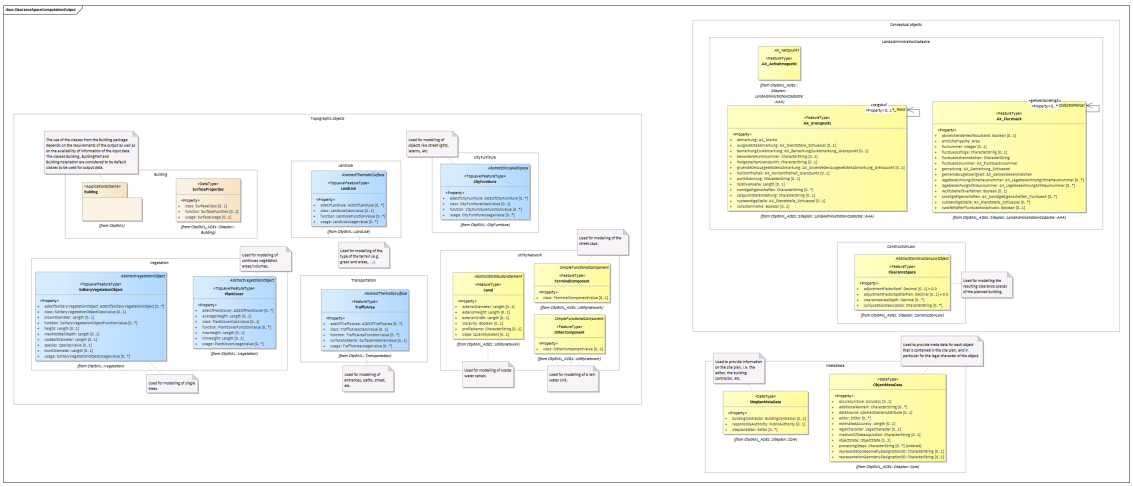


Abbildung 93 CityGML-Profil, welches die Lageplan-Objekte des erstellen Lageplans im CityGML-Format.

5.2.3 Resultat der Erstellung eines 3D-Lageplans

Beschreibung der resultierenden Datei

Das Ergebnis der Ausführung der FME-*Workbench* ist eine valide GML-Datei, gemäß dem CityGML-Profil in Listing 5.1. Der CityGML-Lageplan enthält alle Objekte des ursprünglichen Lageplans im CAD-Format, jedoch hat jedes Objekt neben seiner Geometrie auch noch eine semantische Identität. Die Geometrie der Lageplan-Objekte beschränkt sich aufgrund des räumlichen Modellierungskonzepts des CityGML-Modells nun nicht mehr nur auf 2D-Geometrien, sondern wird um die 3D-Repräsentation erweitert.
Nicht nur das geplante Gebäude ist als drei-dimensionales Modell im Lageplan enthalten, auch die Nachbargebäude wurden als LOD2-Gebäude in den Datensatz integriert. Jede der thematischen Begrenzungsflächen des geplanten Gebäudes referenziert die entsprechenden Abstandsflächen, die aus der selbigen Begrenzungsfläche resultieren. Das geplante Gebäude selbst besitzt einen Abstandsraum, welcher eine Solid-Geometrie aufweist.
Wiederkehrende individuelle topographische Objekte, wie z.B. Bäume, oder Straßenlaternen, sind als Objekte mit impliziter Geometrie enthalten und verringern dadurch die Größe der Datei, da sie die Geometrie nicht jedes Mal neu definieren, sondern nur auf die Referenzgeometrie verweisen.
Flächenhafte Objekte wie Verkehrsflächen, Wiesen und Wasserflächen spiegeln die Geländeoberfläche wieder.

Möglichkeiten der Optimierung der Dateigröße

Mit knapp über 22.6 MB (Megabyte) ist die resultierende Datei des 3D-Lageplans relativ groß, obwohl schon auf das Konzept der impliziten Geometrie für wiederkehrende Objekte zurückgegriffen wurde, welches über Geometrie-Referenzierung statt Instanziierung die Dateigröße bereits reduziert. Die zweite Art von Objekten, welche die Dateigröße wachsen lässt, sind Objekte, dessen Geometrie das Gelände beschreiben.

Sie bilden das triangulierte Gelände mit der GML-Geometrie vom Typ Multi-Flächen (*MultiSurface*) nach. Wie in Kumar, Ledoux und Stoter (2016) beschrieben, führt diese Art der Speicherung von Gelände zu sehr großen Dateien, da Punkte redundant im Datensatz enthalten sind. Durch Vermeidung dieser Redundanz, könnte die Datenmenge um den Faktor 25 reduziert werden (Kumar et al., 2016).
Generell ist die Anzahl der verwendeten Einzelflächen im Falle der Multi-Flächen-Geometrie von der Auflösung des DGMs abhängig. Je niedriger die Auflösung des DGMs, also je größer der Horizontal-Abstand zwischen den Datenpunkten, desto weniger Flächen sind notwendig, um das Gelände zu modellieren. Eine Möglichkeit wäre deshalb, ein DGM mit sehr niedriger Auflösung zu verwenden, um die Dateigröße zu minimieren. Allerdings ist v.a. auf dem Grundstück des Bauvorhabens ein detailliertes Abbild des Geländes von Interesse. Um dies trotzdem zu gewährleisten, kann ein anderer Ansatz verfolgt werden.
Auf dem Grundstück des geplanten Gebäudes wird ein hochauflösendes DGM verwendet, für die großräumige Umgebung jedoch ein eher niedriger auflösendes. Es werden also DGMs verschiedener Auflösung für verschiedene Interessensgebiete herangezogen. Für Gebiete von großem Interesse, wie das Baugrundstück, wird, bspw. für die Zentimeter-genaue Berechnung von Abstandsflächen, ein sehr feines DGM benötigt. Für das Umfeld des Baugrundstücks ist das allerdings nicht notwendig, da keine hoch-genauen Berechnungen durchgeführt werden müssen, sondern das Hauptziel ist, den groben Kontext des Bauvorhabens darzustellen.
Dadurch verringert sich die Anzahl der verwendeten Flächen, die für die Modellierung des Geländes notwendig sind schon einmal deutlich, da die hohe Auflösung nur noch auf dem Baugrundstück selbst vorhanden ist.
Es kann eine grobe Abschätzung über die Reduktion der Dreiecksflächen bei Verwendung von DGMs verschiedener Auflösungen getroffen werden.
Generell wird Anzahl der bei einer Triangulation zu erwartenden Dreiecksflächen über die Formel $2(V-1)-B$ beschrieben (Lee & Lin, 1986). Dabei ist V die Anzahl der Punkte in der Punktmenge und B die Anzahl de Punkte auf der konvexen Hülle von V. Bei einer betrachteten Fläche A mit einer Punktdichte von ρ ergibt sich $V = A \cdot \rho$.
In Abbildung 94 ist links eine Fläche A_0 mit hoher Punktdichte ρ_{high} dargestellt. Rechts wird die Punktdichte, außer im Interessensgebiet A_1, verringert. Die Fläche A_1 weist eine Punktdichte von ρ_{high} auf, die Umgebung eine Dichte von $\rho_{low} = f \cdot \rho_{high}$, welche als faktorisierte Dichte von ρ_{high} dargestellt werden kann.
Die Anzahl Dreiecke in der linken Konfiguration ergibt sich zu $N_{high} = 2(A_0 \cdot \rho_{high} - 1) - B$. Die Anzahl der Punkte auf der konvexen Hülle (gestrichelte Linie um A_0) wird in beiden Konfiguration als konstant angesehen, sodass $B = const.$ gilt.
Um die Punktmenge rechts zu bestimmen, wird die Fläche A_1 von der Fläche A_0 abgezogen. Damit kann über $\rho_{low}(A_0 - A_1) = f \cdot \rho_{high} \cdot (A_0 - A_1)$ die Anzahl der Punkte in der Umgebungsfläche berechnet werden.

In der Fläche A_1 befinden sich $\rho_{high} \cdot A_1$ Punkte, sodass sich für die Gesamtmenge an Dreiecken die Zahl $N_{mix} = 2(\rho_{low}(A_0 - A_1) + \rho_{high} \cdot A_1 - 1) - B$ ergibt. Anzumerken ist, dass diese Form der Betrachtung sehr vereinfacht ist, da durch die Bearbeitung der DGMs mit Schneide-Operationen (vgl. Abschnitt 5.2.2) Artefakte entstehen können, die in diesen Formeln keinen Ausdruck finden.
Auch die Punkte auf der konvexe Hülle des inneren DGMs in Fläche A_1 werden vernachlässigt (die rechte Konfiguration wird als eine Punktmenge mit unterschiedlichen Punktdichten betrachtet), was zu einer eher pessimistischeren Abschätzung führt, da diese Zahl ja in der Formel für die Anzahl der Dreiecke subtrahiert wird.
Aus den zwei berechneten Anzahlen der Dreiecke N_{high} und N_{mix} kann das Verhältnis $\frac{N_{mix}}{N_{high}}$ berechnet werden:

$$\frac{N_{mix}}{N_{high}} = \frac{2(\rho_{low}(A_0 - A_1) + \rho_{high} \cdot A_1 - 1) - B}{2(A_0 \cdot \rho_{high} - 1) - B} \tag{5.1}$$

Die Auswirkungen der Verwendung von zwei DGMs mit unterschiedlicher Auflösung soll an einem Zahlenbeispiel verdeutlicht werden. Für $B = 100$, $A_0 = 1000\,m^2$, $A_1 = 200\,m^2$, sowie $\rho_{high} = \frac{1}{m^2}$ und $\rho_{low} = \frac{0.2}{m^2}$ ist das Verhältnis $\frac{N_{mix}}{N_{high}} = 0.326$. Damit reduziert sich die Anzahl der Dreiecksflächen, im Vergleich zur Verwendung eines hochauflösenden DGMs für die gesamte Fläche A_0, um ca. 68 %.

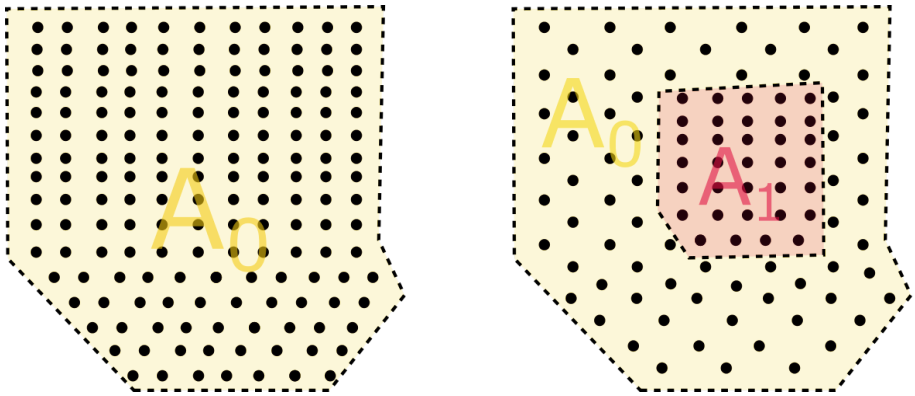


Abbildung 94 Fläche mit DGM mit hoher Punktdichte (links) und Fläche mit zwei DGMs unterschiedlicher Punktdichte (rechts).

Möglichkeiten der Visualisierung

FME Data Inspector

Wird ie Lageplan-Datei in den Safe Software *FME Data Inspector* geladen, stehen dort, neben der Visualisierung des Lageplans, auch die Attribute jedes Objektes in Tabellenform zur Verfügung. Die einzelnen Laeplan-Objekte können in der graphischen Oberfläche selektiert werden. Das ausgewählte Objekt wird dann in der Tabellenansicht markiert und auch die Objekt(/*Feature*)-Informationen werden rechts im Panel angezeigt. Zusätzlich ist es möglich, über den Bereich links in der Benutzeroberfläche bestimmte Objektarten (*Feature Types*) ein- oder auszublenden.

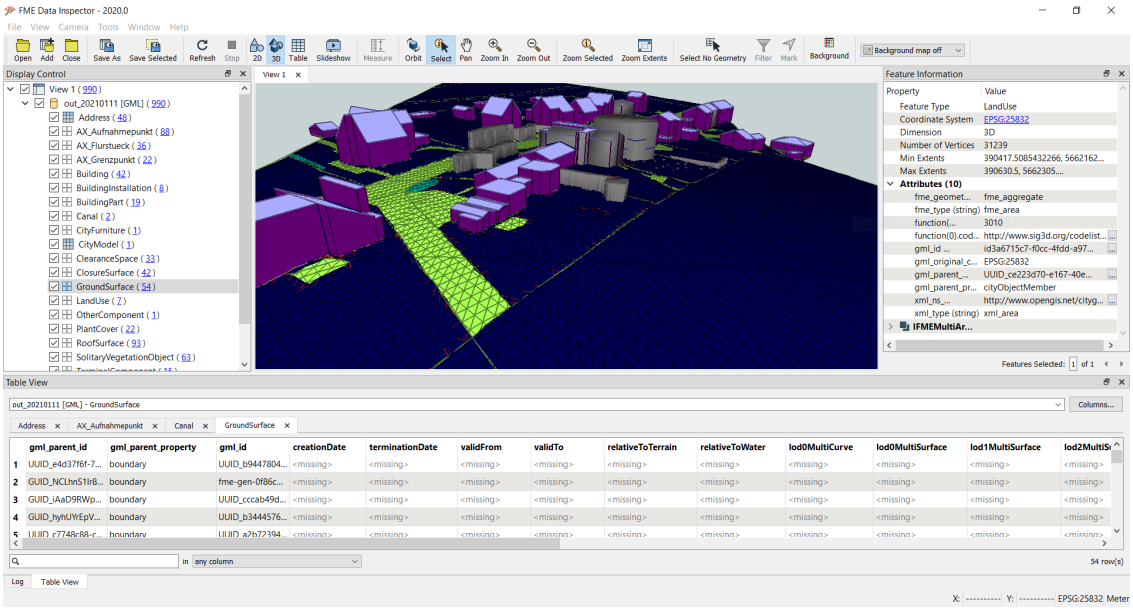


Abbildung 95 Inspektion des 3D-Lageplans im *FME Data Inspector*. In dem Panel links sind die eingelesenen Objektarten (*Feature Types*) zu sehen. Unten in der Tabelle sind die enthaltenen Daten der einzelnen Objekte (*Features*) tabellarisch aufgelistet. Im rechten Bereich sind zusätzlich FME-interne Attribute zu sehen, die bei der Abbildung auf die FME-interne Darstellung entstehen.

X3D

Zu Zwecken der reinen Visualisierung können in FME die Lageplan-Objekte zusätzlich zum GML-Format noch im X3D-Format ausgegeben werden. X3D ist ein Plattform-unabhängiges Datenformat für die Visualisierung von 3D-Szenen und -Objekten (Web3D Consortium, o. J.). X3D-Dateien können bspw. in einem HTML5- und WebGL-fähigen Browser mittels der JavaScript-Bibliotheken von X3DOM (`https://www.x3dom.org/`) dargestellt werden.
In FME wird je Objektart (*Feature Type*) eine X3D-Datei geschrieben, die Geometrie und Parameter der Erscheinung der Objektart enthält. Die X3D-Dateien können anschließend in ein HTML-Skelett eingebettet werden (siehe *index.html*). Die X3D-Objekte stehen zur Laufzeit im DOM zur Verfügung und können daher über JavaScript dynamisch manipuliert werden.
Neben der Vielfalt an Möglichkeiten der Visualisierung und Manipulation der 3D-Objekte ist ein großer Nachteil allerdings, dass sich die Anwendung von X3D im Kontext von CityGML auf die Visualisierung beschränkt. Es steht, außer der Objekt-Erscheinung, keine semantischen Information der Objekte aus dem ursprünglichen CityGML-Modell zur Verfügung.

```
<!DOCTYPE html>
<html>
    <head>
        <script type='text/javascript' src='http://www.x3dom.org/download/x3dom.js'> </script>
        <link rel='stylesheet' type='text/css' href='http://www.x3dom.org/download/x3dom.css'></link>
    </head>
    <body>
        <x3d id="the_x3delement">
            <scene id="scene">
                <inline id="AX_Flurstueck" nameSpaceName="AX_Flurstueck" mapDEFToID="true" url="data/AX_Flurstueck.x3d"></
      inline>
                <inline id="AX_Grenzpunkt" nameSpaceName="AX_Grenzpunkt" mapDEFToID="true" url="data/AX_Grenzpunkt.x3d"></
      inline>
                <inline id="AX_Aufnahmepunkt" nameSpaceName="AX_Aufnahmepunkt" mapDEFToID="true" url="data/
      AX_Aufnahmepunkt.x3d"></inline>
                <inline id="LandUse_Forest" nameSpaceName="LandUse_Forest" mapDEFToID="true" url="data/LandUse_Forest.x3d"
      ></inline>
                <inline id="LandUse_Water" nameSpaceName="LandUse_Water" mapDEFToID="true" url="data/LandUse_Water.x3d"></
      inline>
                <inline id="LandUse_Grassland" nameSpaceName="LandUse_Grassland" mapDEFToID="true" url="data/
      LandUse_Grassland.x3d"></inline>
                <inline id="RoofSurface" nameSpaceName="RoofSurface" mapDEFToID="true" url="data/RoofSurface.x3d"></inline
      >
                <inline id="WallSurface" nameSpaceName="WallSurface" mapDEFToID="true" url="data/WallSurface.x3d"></inline
      >
                <inline id="GroundSurface" nameSpaceName="GroundSurface" mapDEFToID="true" url="data/GroundSurface.x3d"></
      inline>
                <inline id="BuildingInstallation" nameSpaceName="BuildingInstallation" mapDEFToID="true" url="data/
      BuildingInstallation.x3d"></inline>
                <inline id="TrafficArea" nameSpaceName="TrafficArea" mapDEFToID="true" url="data/TrafficArea.x3d"></inline
      >
                <inline id="TerminalComponent" nameSpaceName="TerminalComponent" mapDEFToID="true" url="data/
      TerminalComponent.x3d"></inline>
                <inline id="SolitaryVegetationObject" nameSpaceName="SolitaryVegetationObject" mapDEFToID="true" url="data
      /SolitaryVegetationObject.x3d"></inline>
                <inline id="Canal" nameSpaceName="Canal" mapDEFToID="true" url="data/Canal.x3d"></inline>
                <inline id="CityFurniture" nameSpaceName="CityFurniture" mapDEFToID="true" url="data/CityFurniture.x3d"></
      inline>
                <inline id="ClearanceSpace_Hull" nameSpaceName="ClearanceSpace_Hull" mapDEFToID="true" url="data/
      ClearanceSpace_Hull.x3d"></inline>
                <inline id="ClearanceSpace_Surfaces" nameSpaceName="ClearanceSpace_Surfaces" mapDEFToID="true" url="data/
      ClearanceSpace_Surfaces.x3d"></inline>
                <inline id="OtherComponent" nameSpaceName="OtherComponent" mapDEFToID="true" url="data/OtherComponent.x3d"
      ></inline>
                <inline id="PlantCover" nameSpaceName="PlantCover" mapDEFToID="true" url="data/PlantCover.x3d"></inline>
            </scene>
        </x3d>
    </body>
</html>
```

Listing 5.2: index.html

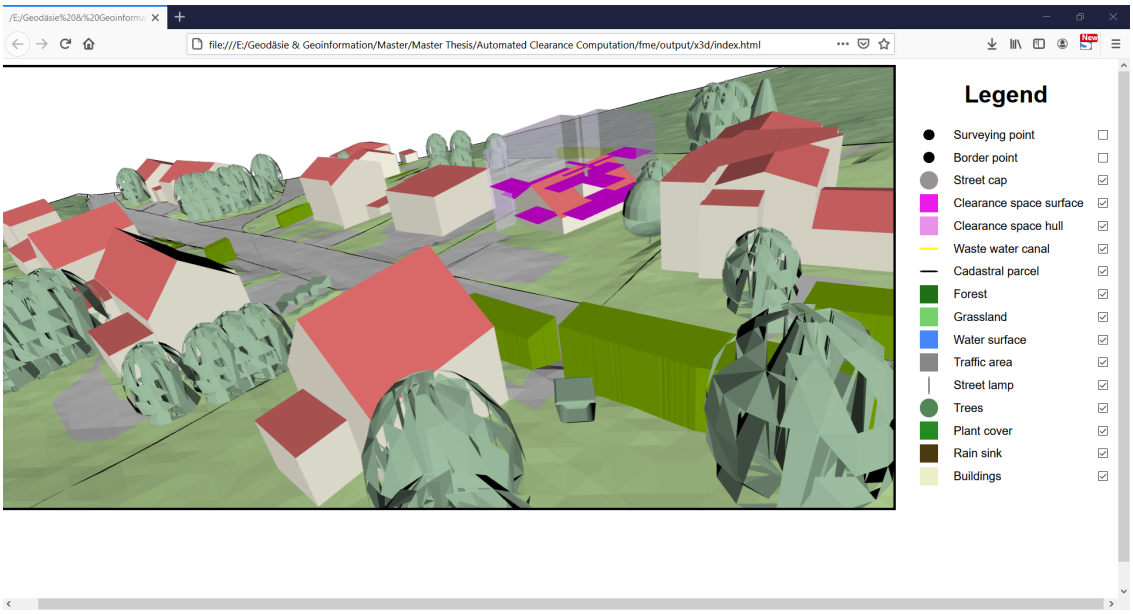


Abbildung 96 Visualisierung des Lageplans im Browser. Über JavaScript-Programmierung können die 3D-Objekte dynamisch manipuliert werden, wie hier z.B. das Ein- und Ausblenden über das Setzen von Häkchen in der Legende.

Hinzufügen einer digitalen Signatur

In Unterabschnitt 3.1.3 wurde bereits auf das Konzept der digitalen Signatur von Dokumenten eingegangen. Hier soll nun anhand eines kleinen Beispiels (siehe Listing 5.3) die Vorgehensweise des Signierens eines XML-Dokuments geschildert werden, welche analog auf die Lageplan-Datei anwendbar ist.

```
<song>
  <title>random title</title>
  <artist>random artist</artist>
  <duration>4:14</duration>
</song>
```

Listing 5.3: document.xml

Für Testzwecke wird zunächst mittels des Open-Source-Tools *openssl* (`https://www.openssl.org/`) ein privater Schlüssel, sowie ein sogenanntes *self-signed* Zertifikat erstellt. Das Zertifikat enthält, neben weiteren Angaben, den öffentlichen Schlüssel.

```
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
```

Listing 5.4: createCertificate.bat
(`https://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/com.ibm.apic.cmc.doc/task_apionprem_gernerate_self_signed_openSSL.html`)

Für das Schreiben der Signatur in das XML-Dokument wird das Python-Paket *SignXML* (`https://signxml.readthedocs.io/en/latest/`) verwendet. Neben *SignXML* exisitert außerdem die Python-Einbindung der *XML Security Library* (`https://pythonhosted.org/xmlsec/index.html`). Der folgende Python-Quellcode zeigt, wie die Signatur des Dokuments mittels des Pakets *SignXML* bewerkstelligt werden kann.

```
# -*- coding: utf-8 -*-

from lxml import etree;
from signxml import XMLSigner, XMLVerifier;

xmlDoc = "./document.xml";
sslCertificate = "./certificate.pem";
sslKeys = "./key.pem";

# Read SSL files
key = None;
cert = None;

with open(sslCertificate) as f:
    cert = f.read();

with open(sslKeys) as f:
    key = f.read();

# Parse XML file and extract root from DOM
xmlRoot = etree.parse(xmlDoc).getroot();

# Sign the document
signedRoot = XMLSigner().sign(xmlRoot, key=key, cert=cert); # cert=cert is not obligatory

xmlSigned = etree.tostring(signedRoot);

# Write out the signed document
with open("./document_signed.xml", "wb") as f:
    f.write(xmlSigned);
```

Listing 5.5: xmlSignature.py

Das resultierende signierte XML-Dokument ist in *Listing 5.6* zu sehen. Die Methode der Verknüpfung der Signatur mit den Daten ist *enveloped*. Das Signatur-Element ist also im betroffenen Element als Kind-Element enthalten (vgl. Unterabschnitt 3.1.3). Definitionen der einzelnen Kind-Elemente von *<ds:Signature>* sind in der entsprechenden W3C-*Recommendation* (`https://www.w3.org/TR/xmldsig-core1/`) einzusehen.

```
<song>
  <title>random title</title>
  <artist>random artist</artist>
  <duration>4:14</duration>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>nbNw/RuNO1Qufg8LRLYYBomRfTLzg68+N8gmUUw1p8M=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>YutTbXO8Z9FZYLAfhP9Ho417sYIwjL826zxFqlgMfGmE/X933KQA89LALZN55UjhXNlC0nlJD8gMfUdp1Ob7+CRdrK9lkhIX+
      lazm2z0pD46B+jwZQClZrMcFHNBavKLZNn9yHXPAPQlH7juwpuM2K/9zmTecyK/bkqYMRcLGfj/
      s7GIJMsHxouftx1Ydv9GP1H2NcBoYGkky7rJ6ZgH0TNeY+1L6lQwj0hWwrKNadPlkBg/tCnInpotOsNQJhEUjp1ViRizxChPEqo/
      wTxu3wJLoDahhUsjBFzjKQkzcN8S+oJjIpVaTj/fkK16Rq6UtD5EDv+JYdQlG46fkpj4yQ==</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>MIIDazCCAlOgAwIBAgIUCO0TAlmSUTc27BVNLUnDqWT3ilcwDQYJKoZIhvcNAQEL
BQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUxITAfBgNVBAoM
GEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDAeFw0yMTAxMTYxNzEwMzhaFw0yMjAx
MTYxNzEwMzhaMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb21lLVN0YXRlMSEw
HwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggEiMA0GCSqGSIb3DQEB
AQUAA4IBDwAwggEKAoIBAQC60ria1VrhjOtE2I2+PVq26mN6zmMcC8c1dk21SpCl
foNowsD0PeCeOWyY8CiOJCC3WgkOycjkBtSEdnBn8yGWVX3UuaHLorZmPwjntrVt
1PoT/0wEwUokcrrgX0xP7ajDYB7S8L1DOqjx/nGpJ345UBYDt9CMjqicUnMWY1JU
GubKT4OA2sOO7EWuF1O3oeuDWXDkJUl7Bncax7C6U265feg/Rp2mB5TXPaBssIsB
zxAX9TbPDqdBe+F+dINa6/poOm0zSTEBBCxVqo/UkQjf+2FEDR4t+jJdzyj/qbBc
aC4s5rnkzDspNCYu7eWi+2F/BmmMlTED1aKeJK8pJDWRAgMBAAGjUzBRMB0GA1Ud
DgQWBBQtxpghsLgooKiP1OkEc6UTkTtfszAfBgNVHSMEGDAWgBQtxpghsLgooKiP
1OkEc6UTkTtfszAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQAg
1wYBk/HzJ8GpQvPxgEsRy1ta5BwyUFCV5nGJwCu0gEYq7oZM1E+L8swNpda5yUnf
c/NjMArh373tXNASdxCzs6M0JnIB3pnYRCFuL4AsQ/YjLGPYyJwkN+qI1M6GaCCl
g8okrzazDtZPAQciCg0/DQ5QuSWrdcttnhuzxK/RQzMznqrclSzwNpFhL+hS91Sr
wG8ox3moWMZ/tAn2I3iRV7FIZQmJGASeQG9tVPFsl9+I6I6012rAVL7jMaU7FmAf
yoNTAmNYTcg2zu6jRAOpyej5k4g5UEtjOQpnO5o/6gJvIlZrTc+7BKepViianb+e
WzxokEUQI5D4kW4rMpGq
    </ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
</song>
```

Listing 5.6: document_signed.xml

Wird das signierte Dokument document_signed.xml verifiziert, so ist das erwartete Zertifikat der Dokumentenautors notwendig, um eine erfolgreiche Validierung zu erreichen. Ist zu beachten, dass zumindest in der Implementierung von *SignXML* u.a. die Kanonisierung (*canonicalization*) des XML-Dokuments bei der Verifizierung zu beachten ist (siehe `https://technotes.shemyak.com/posts/xml-signatures-with-python-elementtree/`).
In Listing 5.7 ist eine mögliche Implementierung für die Verifizierung eines XML-Dokuments illustriert.

```
# -*- coding: utf-8 -*-

from lxml import etree;
from signxml import XMLVerifier;

xmlDoc = "./document_signed.xml";
sslCertificate = "./certificate.pem";

# Read SSL files
cert = None;

with open(sslCertificate) as f:
    cert = f.read();

# Read XML document
xmlRoot = etree.parse(xmlDoc).getroot();

try:
    verifiedXML = XMLVerifier().verify(xmlRoot, x509_cert=cert).signed_xml;
    print("Verification was succesfull.");
except:
    print("Verification failed.");
```

Listing 5.7: xmlVerification.py

6 Diskussion

6.1 Diskussion der Forschungsfragen

6.1.1 Datenmodell eines Lageplans in 3D

Im Datenmodell zu modellierende Objekte

Die Analyse der Eingangsdaten in Unterabschnitt 3.1.1 für die Erstellung eines Lageplans lässt die Aussage zu, dass keine statische oder endliche Objektmenge für den Lageplan existiert. Prinzipiell ist der Inhalt eines (amtlichen) Lageplans im entsprechenden Paragraphen der BauPrüfVO definiert. Dort sind zwar Beschreibungen der Objektarten zu finden, jedoch sind diese Beschreibungen generell gehalten und es werden oft nur Sammelbegriffe für bestimmte Objektarten verwendet.
Der Lageplan soll relevante Topographie (unter und über der Erdoberfläche), sowie soziale und rechtliche Konstrukte beinhalten. Beide Domänen sind jedoch zeitlichen Entwicklungen und Änderungen in Gesetzen, Technologien, Stadtentwicklung und im sozialen Bereich unterworfen, sodass keine abschließende Objektmenge definiert werden kann.
Allein aufgrund der Datenherkunft einzelner Objektuntermengen, kann diese Untermenge auf die im Ursprungsdatenmodell definierten Objekte klar eingeschränkt werden. Dies trifft bspw. für die Grundlage des Lageplans, den Auszug aus dem Kataster, zu. Hier kann sich auf das INSPIRE-Datenmodell zum Flurstück, oder das AAA-Datenmodell der AdV bezogen werden. Auch die Objektmenge des Bebauungsplans ist über vorhandene Datenmodelle aus der INSPIRE-Initiative (*Planned Land Use*) oder dem XPlanung-Standard klar definiert. Doch auch hier gilt, dass die Objektmengen bzw. der Datenmodelle nicht statisch sind, sondern laufenden Weiterentwicklungen und Anpassungen ausgesetzt sind.
Neben dem Anwendungsfall des Bauantrags existieren zahlreiche weitere Anwendungsfälle für Lagepläne (siehe Unterabschnitt 3.1.3). Lagepläne dienen bspw. auch bei der Ermittlung von Immobilienwerten oder Versicherungsbeiträgen. Dies führt dazu, dass u.a. auch physikalische Eigenschaften von topographischen Objekten, oder Elemente der Risiko-Abschätzung im Datenmodell enthalten sein müssen.
Die dynamische Objektmenge macht einen grundsätzlichen Vorteil des CityGML-Datenmodells sichtbar. Die generische Objektklassen des CityGML-Standards können bei der Erstellung eines Lageplans immer dann zum Einsatz kommen, wenn ein Objekt nicht auf eine vorhandene Klasse abgebildet werden kann.

Thematische Aufteilung der Lageplan-Objekte

Die Lageplan-Objekte lassen sich prinzipiell in zwei Domänen einteilen. Topographische Objekte weisen Merkmale und Objekte physikalischer Natur auf, wohingegen konzeptionelle Objekte virtuelle Objekte mit sozialem und konzeptionellen Aspekt sind, ohne direkte physikalische Eigenschaften.
Aus den Anwendungsfällen des Lageplans in Unterabschnitt 3.1.3 gehen allerdings weitere notwendige Aufteilungen der Lageplan-Objekte hervor. Diese Aufteilungen beziehen auf die Analyse bestimmter Objektuntermengen der im Lageplan enthaltenen Gesamtmenge an Objekten.
Deshalb wird eine zusätzliche Unterteilung in geplante und existierende Objekte eingeführt. Es entstehen somit vier Schnittmengen, welche im Datenmodell der SitenplanADE noch weiter spezifiziert werden:

- Existierende topographische Objekte
- Existierende konzeptionelle Objekte
- Geplante topographische Objekte
- Geplante konzeptionelle Objekte

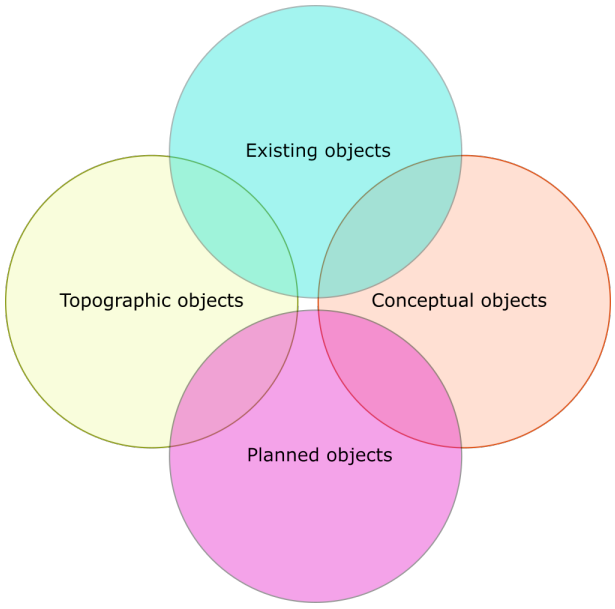


Abbildung 97 Die Einteilung der Lageplan-Objekte in die topographische und konzeptionelle Domäne, sowie in deren Zustand (existent oder geplant), resultiert in vier Schnittmengen.

Wahl der Geometrie der Lageplan-Objekte

Die Geometrie, sowie das LOD, welches für die Darstellung der Lageplan-Objekte verwendet wird, ist abhängig von den verfügbaren Eingangsdaten und der gewünschten Dimension des Lageplans. Es ist also keine bestimmte Geometrie für Lageplan-Objekte obligatorisch. Grundsätzlich sollten alle Lageplan-Objekte in 3D-Lageplänen 3D-Geometrien aufweisen, in 2D-Lageplänen dementsprechend 2D-Geometrien. Das LOD bzgl. der Geometrie sollte in Anbetracht der Dateigröße so gewählt werden, dass so wenige Details wie nötig, aber so viele wie notwendig, bereitgestellt werden. Das bedeutet, dass v.a. in der nahen Umgebung (bspw. das Baugrundstück) möglichst hohe LODs verwendet werden sollten. In der großräumigen Umgebung würde jedoch der Einsatz geringerer LODs ausreichen.

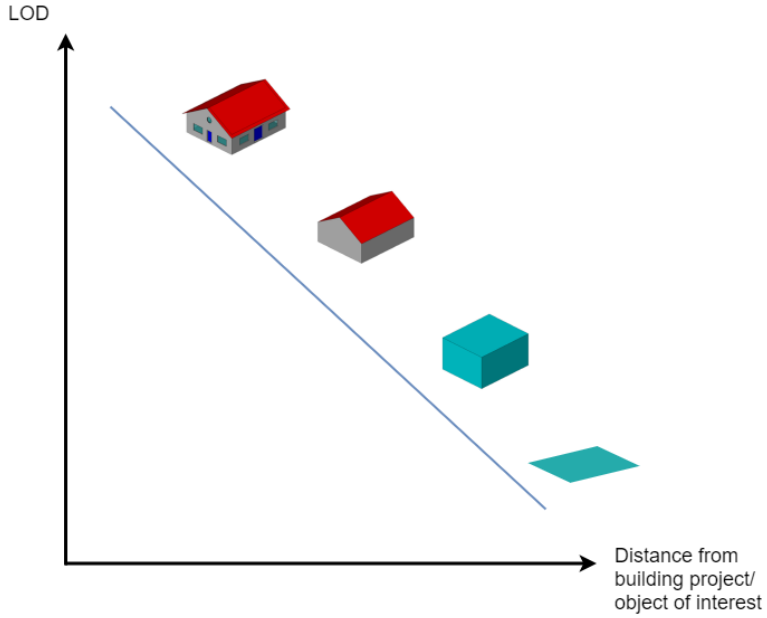


Abbildung 98 Abnahme des benötigten LODs in Abhängigkeit zur Distanz zum Bauprojekt bzw. zum Objekt des Interesses (Quelle des Gebäudes in verschiedenen LODs: (Kolbe et al., 2020)).

Verschiedene Varianten des Lageplans

Werden verschieden Versionen eines Lageplans mit unterschiedlichen thematischen Objektmengen benötigt, so gibt es verschiedene Möglichkeiten dies zu bewerkstelligen:

- **Erstellung eines CityGML-Lageplan-Profils**: Bestimmte Module und Klassen werden ausgeschlossen, welche nicht im Lageplan erwünscht sind. Instanz-Dokumente können dann gegen die Schema-Dateien des Profils validiert werden.
- **Beschreibung durch Text**: Die Variante des Lageplans wird als Fließtext definiert. Es liegt an dem*der Ersteller*in, sich an die Vorgaben zu halten. Eine Schema-Validierung des Instanz-Dokuments ist nicht möglich.

Ableitung eines 2D-Lageplans vom 3D-Lageplan

Eine Ableitung eines 2D-Lageplans von einem 3D-Lageplan im CityGML-Format kann bspw. in einer FME durchgeführt werden. In der Prozessierung muss von allen Objekten die z-Komponente entfernt werden. Dadurch werden alle 3D-Geometrien auf zwei Dimensionen reduziert. Eine Schwierigkeit besteht darin, valide Geometrien zu erzeugen, da z.B. sich schneidende Polygon oder Multi-Geometrien entstehen können, wenn ein Volumenkörper auf eine 2D-Ebene projiziert wird.

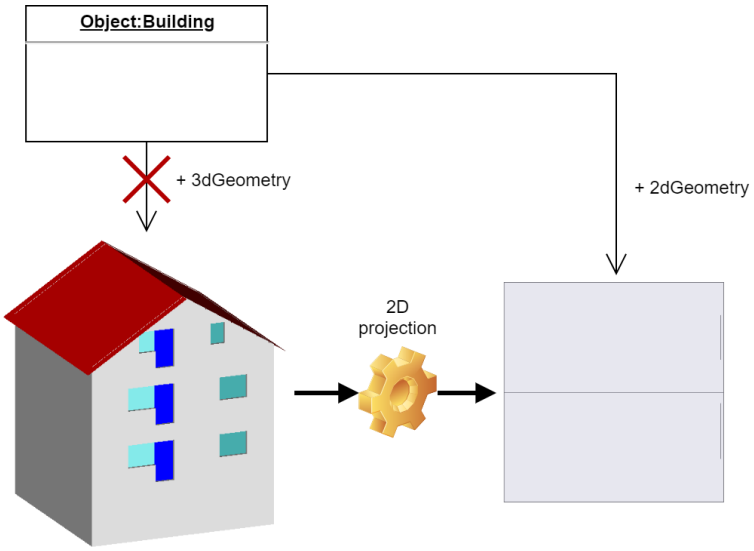


Abbildung 99 Die 3D-Geometrie des Gebäude-Objekts wird in der FME dereferenziert und in eine 2D-Geometrie transformiert.

Alternativ kann den Lageplan-Objekten die 2D-Geometrie als zusätzliche Geometrie niedrigeren LODs im Datensatz hinzugefügt werden. Das Datenmodell der SiteplanADE bietet in der ADE-Klasse *ObjectMetaData* das Attribut ”representationGeometryDesignation2D”, welches den Attribut-Bezeichner der 2D-Geometrie enthält. Eine Software könnte dann zunächst dieses Attribut auslesen, und basierend auf dem Bezeichner die 2D-Geometrie des Lageplan-Objekts extrahieren.

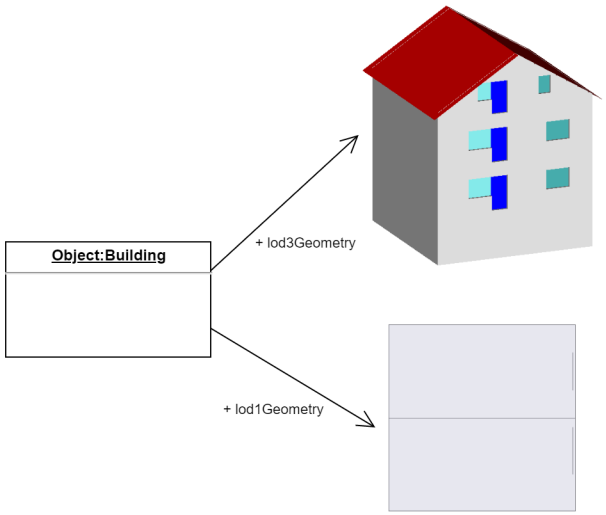


Abbildung 100 Das Gebäude-Objekt wurde mit 3D- und 2D-Geometrie instanziiert. Für die Erstellung eines 2D-Lageplans wird in einer Software die 2D-Geometrie extrahiert.

Elektronische Signatur des XML-Dokuments

Für die elektronische Beurkundung wird die qualifizierte elektronische Signatur als sinnvoll erachtet. Nach ZPO (Zivilprozessordnung) §371a Absatz 3 gilt:

> *„Auf elektronische Dokumente, die von einer öffentlichen Behörde innerhalb der Grenzen ihrer Amtsbefugnisse oder von einer mit öffentlichem Glauben versehenen Person innerhalb des ihr zugewiesenen Geschäftskreises in der vorgeschriebenen Form erstellt worden sind (öffentliche elektronische Dokumente),* ***finden die Vorschriften über die Beweiskraft öffentlicher Urkunden entsprechende Anwendung****.* ***Ist das Dokument von der erstellenden öffentlichen Behörde oder von der mit öffentlichem Glauben versehenen Person mit einer qualifizierten elektronischen Signatur versehen, gilt § 437*** *entsprechend."*

Die Vorschriften über die Beweiskraft öffentlicher Urkunden sind in ZPO §415 geregelt. ZPO §415 Absatz 1 sagt aus, dass die öffentliche Urkunden als volles Beweismittel angesehen werden kann (*„Urkunden [...] begründen [...] vollen Beweis des durch die Behörde oder die Urkundsperson beurkundeten Vorganges."*).
ZPO §437 Absatz 1 bestätigt die Echtheitsvermutung für Urkunden von öffentlichen Behörden oder öffentlich beglaubigte Personen.
Demnach sollte die qualifizierte elektronische Signatur für die Beurkundung von digitalen XML-Lageplänen geeignet sein und auch in Gerichtsverfahren als Beweismittel im zugelassen werden.

6.1.2 Automatisierte Abstandsflächenberechnung

Erforderliche Eingangsdaten

Für die Berechnung sind die in Unterabschnitt 4.1.2 festgestellten Eingangsdaten notwendig:

- 3D-Gebäudemodell
- Gebäudeklasse und überirdischen Vollgeschosse
- Ankerpunkt und Rotationswinkel
- DGM
- Mathematische Berechnungsregeln und Parameter nach BauO
- Grundstücksgrenzen (aus Kataster)
- Art der baulichen Nutzung des Baugebiets des Gebäudes (aus Bebauungsplan)
- Öffentliche Verkehrs-, Grün- und Wasserflächen (aus Bebauungsplan)

Je nachdem in welcher Entwicklungsumgebung bzw. Software der Algorithmus zur Abstandsflächenberechnung implementiert werden soll, ist die Frage nach dem Datenformat der Eingangsdaten unterschiedlich zu beantworten.
Generell wird für das Gebäudemodell das CityGML-Format wegen seines geometrischen Modellierungsansatzes und seiner Semantik als beste Option eingeschätzt. Für die restlichen Parameter ist das Datenformat von den Möglichkeiten der Werkzeuge und Bibliotheken, die in der verwendeten Software bereitstehen, abhängig. Es können aber, wie im Fall der Berechnungsregeln oder des Ankerpunkts/Rotationswinkels, Datenstrukturen vorgegeben werden.
Für Objekte mit geometrischen Eigenschaften kann zudem die erwartete bzw. geforderte Geometrie definiert werden.
Ein weiterer Ansatz ist, alle Eingangsdaten in einem gemeinsamen Datensatz, z.B. im CityGML-Format, zusammenzufassen. Das hat den Vorteil, dass die Software nur mit einem Datenformat umgehen muss. Das CityGML-Format würde sich sehr gut für diesen Ansatz eignen, da die topographischen Klassen bereits im Standard enthalten sind.

Level of Detail des Gebäudemodells für die Abstandsflächenberechnung

Aus den im Gesetzestext in BauO NRW §6 zur Berechnung der Abstandsflächen definierten Berechnungsregeln geht hervor, dass neben äußeren Wand- und Dachflächen die Modellierung von Gebäudeinstallationen, Vorbauten und Gesimse erforderlich sind. Dies entspricht einem architektonischem Gebäude-Modell. Trotz der Anforderung an hohen Detailgrad der Modellierung des Gebäudes, aber auch der Gebäudeinstallationen, bringt dies auch einige Nachteile mit sich.

Findet die Berechnung der Abstandsflächen auf Ebene der Polygone der Multi-Flächen-Geometrien der Gebäudekomponenten statt, werden für alle Einzelflächen der Multi-Flächen-Geometrie Abstandsflächen berechnet. Dies ist v.a. bei hoch-detaillierten architektonisch modellierten Gebäudeinstallationen problematisch.

Zur Veranschaulichung soll dies an einem Beispiel verdeutlicht werden. In Abbildung 101 links oben ist ein Gebäudemodell mit Balkonen dargestellt. Die Balkongeländer sind architektonisch modelliert, d.h. es findet keine Generalisierung auf die die Einzelflächen einschließende konvexe bzw. konkave Hülle statt, sondern jede einzelne Geländerkomponente wird modelliert. Kann der Algorithmus zur Abstandsflächenberechnung nun nicht erkennen, dass sich diese Einzelkomponenten innerhalb der geometrischen Hülle des Geländers befinden, wird für jede der Einzelflächen eine Abstandsfläche berechnet. Das Ergebnis ist in Abbildung 101 rechts oben zu sehen. Jede einzelne Geländer-Strebe resultiert in einer eigenen Abstandsfläche. Dies erhöht zum einen Rechenkosten, zum anderen reduziert es die Übersichtlichkeit. Ausreichend wäre es, wenn für die die Geländer-Streben einschließende geometrische Hülle die Abstandsflächen berechnet werden würden, wie es in Abbildung 101 unten in der Mitte gezeigt wird.

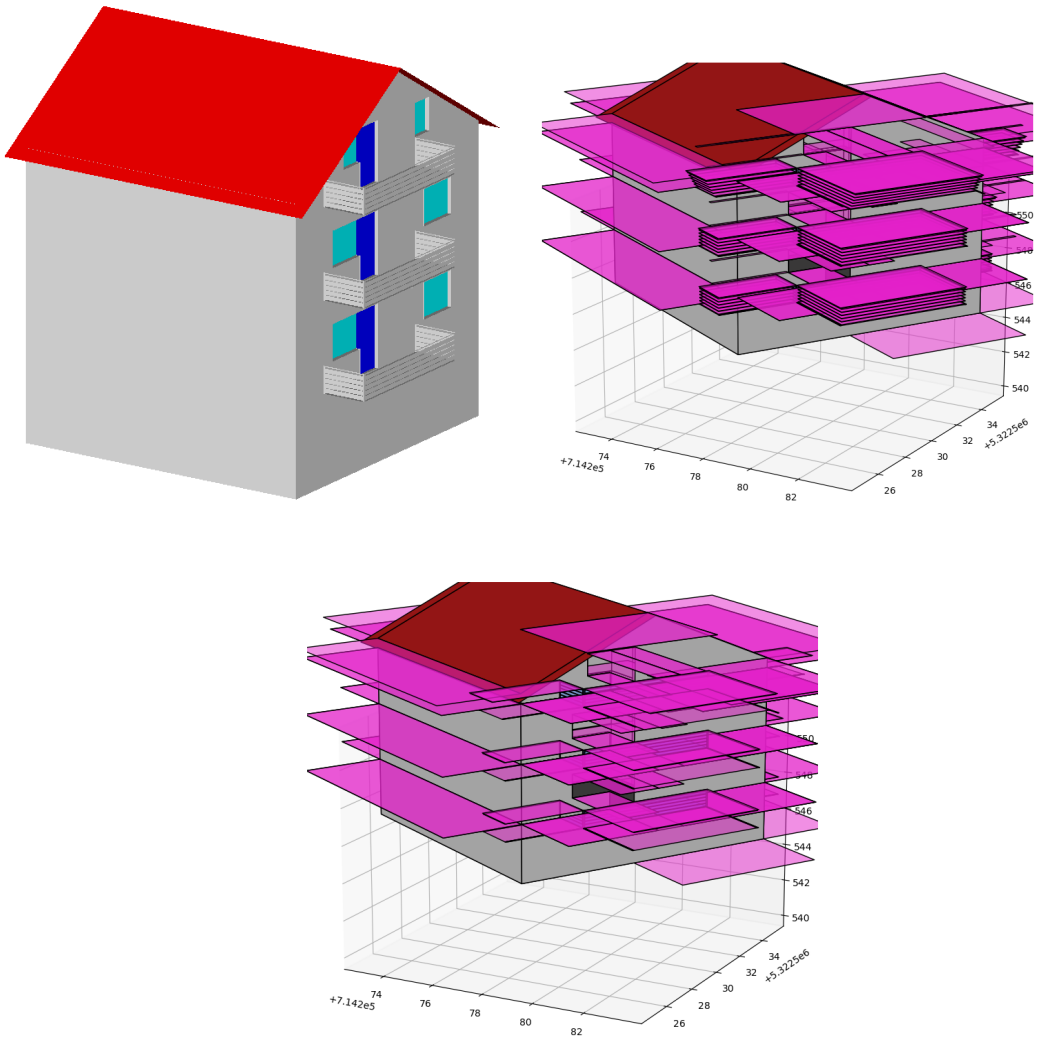


Abbildung 101 Gebäudemodell, mit detaillierter Balkonmodellierung (oben links), die resultierenden Abstandsflächen für jede einzelne Fläche der Balkongeländer (oben rechts) und die resultierenden Abstandsflächen für die geometrische Hüllen der Balkongeländer (unten Mitte).

Die vorigen Ausführungen zeigen, dass es, obwohl die Abstandsflächenberechnung ein Gebäudemodell in hohem Detaillierungsgrad erfordert, in manchen Fällen sinnvoll ist, bestimmte Komponenten (v.a. Gebäudeinstallationen) in einer generalisierten Repräsentation bereitzustellen. Bzgl. des vorigen Beispiels könnten die Balkongeländer, ohne eine Änderung des Gesamteindrucks des Gebäudes oder der resultierenden Abstandsflächen zu bewirken, als eine einzige Fläche (geometrische Hülle des Geländers) pro Seite dargestellt werden.

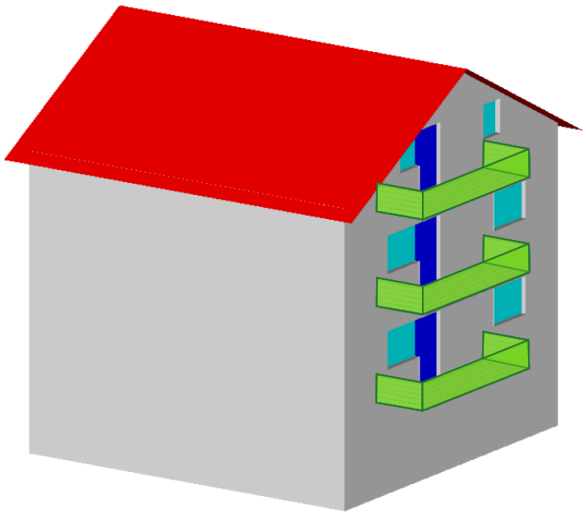


Abbildung 102 Generalisierte Modellierung der Balkongeländer durch geometrische Hüllen der Einzelflächen (grün).

Geometrische Repräsentation des Gebäudemodells für die Abstandsflächenberechnung

Für die geometrische Modellierung des Gebäudemodells stehen zwei Ansätze zur Verfügung. Der aus dem BIM-Bereich bekannte Ansatz mit volumetrischen Körpern, die über Parameter, CSG oder *Sweep* und boolesche Verknüpfung von geometrischen Primitiven beschrieben werden (Kaden et al., 2019). Zum anderen die sogenannte B-Rep aus dem GIS-Bereich, die ein Objekt durch dessen äußeren Begrenzungsflächen repräsentiert (Kaden et al., 2019).
Wie in Unterabschnitt 4.1.2 beschrieben, ist es für die Berechnung der Abstandsflächen erforderlich, die äußeren Begrenzungsflächen aus dem Gebäudemodell zu extrahieren. Da der volumetrische Ansatz aus ganzen Körpern, welche durch Parameter beschrieben werden, besteht, ist der Aufwand aus dieser Repräsentation die Begrenzungsflächen zu extrahieren größer, als in der B-Rep (Donkers et al., 2015; El-Mekawy & Östman, 2010).
Wenn z.B. eine Wand als volumetrisches Objekt mit sechs Flächen modelliert wird, ist außerdem zu bestimmen, welche der Flächen die äußere Begrenzungsfläche des Gebäudemodells beschreibt, da keine semantische Unterscheidung zwischen Innen- und Außenwandflächen vorhanden ist (Donkers et al., 2015; El-Mekawy & Östman, 2010). Bei der B-Rep liegen die Begrenzungsflächen explizit mit Koordinaten vor und bedürfen keiner weiteren Bearbeitung vor der Extraktion. Im Falle von CityGML liegt eine Außenwand bereits identifizierbar als semantisches Objekt mit Flächengeometrie vor.
Die Verwendung von CityGML-Gebäudemodellen erspart demnach aufgrund der geometrischen Darstellung durch die B-Rep und die semantische Unterscheidung von Innen- und Außenflächen ein eine aufwändige Vorverarbeitung des Gebäudemodells, wie es bei Gebäudemodellen vieler BIM-Formaten durch den volumetrischen Ansatz bei der geometrischen Modellierung der Fall wäre.

6.2 Diskussion der Ziele der Arbeit

6.2.1 SiteplanADE

Zusammenfassung der Ergebnisse

Die SiteplanADE stellt eine Erweiterung des CityGML-Modells bzgl. der konzeptionellen Domäne in den Bereichen Landentwicklung und Landadministration/Kataster dar. Der Kontext der Erstellung dieser ADE sind die Entwicklungen hin zur Digitalisierung und Automatisierung im Bauantragsverfahren (vgl. Abschnitt 2.2). Die SiteplanADE ermöglicht es, einen digitalen Lageplan im CityGML-Format zu erstellen, der für Maschinen interpretierbar ist und eine (teil-)automatisierte Verarbeitung und Prüfung erlaubt.
Aufgrund der internationalen Reichweite der Digitalisierung und Automatisierung im Bereich des Bauantrags, berücksichtigt das Datenmodell der SiteplanADE existierende internationale Standards, wie INSPIRE und LADM. Kompatibilität zu den deutschen nationalen Standards wird durch Einbindung und teilweise Übernahme der Modelle XPlanung (Bebauungsplan) und AAA (Liegenschaftskataster) geschaffen.
Auch Klassen bzgl. anderer Anwendungsfälle eines Lageplans, wie z.B. die Ermittlung von Versicherungsbeiträgen oder des Werts einer Immobilie, finden sich in der SiteplanADE wieder.

Fazit

Durch die Einbindung der nationalen und dadurch sehr detailreichen Standards XPlanung und AAA, wird das Datenmodell der SiteplanADE sehr groß und unübersichtlich. In der Praxis wird für die Erstellung eines Lageplans mit der SiteplanADE nur ein kleiner Ausschnitt des gesamten Modells benötigt. Trotzdem ist es notwendig, alle Klassen der Modelle beizubehalten, da XPlanung (Modell für digitalen Bebauungsplan) und AAA (Modell für die Katasterführung nach VermKatG) auf gesetzlicher Basis beruhen (K.-U. Krause, o. J.).
In Abschnitt 5.2.2 ist das CityGML-Profil des Lageplans, der im Zuge dieser Arbeit erstellt wurde dargestellt. Dieses Profil mach deutlich, dass nur ein geringer Teil des gesamten CityGML-Modells und der SiteplanADE verwendet wurde.
Trotzdem ist es sinnvoll, alle Module im CityGML-Profil aus Abschnitt 5.2.2 zu behalten, um flexibel Lagepläne mit unterschiedlichen Objektmengen erstellen zu können.

6.2.2 Automatisierte Abstandsflächenberechnung

Zusammenfassung der Ergebnisse

Der Abstandsflächenalgorithmus, welcher in dieser Arbeit entwickelt wurde, führt die Berechnungen anhand eines Gebäudemodells im CityGML-Format durch. Für die Berechnung sind einige Attribute notwendig, die standardmäßig nicht im CityGML-Modell für Gebäude enthalten sind. Es ist daher die Entwicklung einer ADE notwendig, die es ermöglicht, die erforderlichen Attribute im Gebäudemodell bereitzustellen (siehe Unterabschnitt 4.2.2). Die zusätzlichen Attribute spezifizieren vornehmlich die Relevanz und Rolle der Begrenzungsflächen und Gebäudeinstallationen im Kontext der Abstandsflächenberechnung. Auch notwendige topologische Relationen zwischen Objekten werden über die ADE möglich.
Im Abstandsflächenalgorithmus werden die in BauO NRW §6 definierten Regeln, Bedingungen und Parameter implementiert. Die Bedingungen, ob eine Abstandsfläche aus einer Gebäudekomponente resultiert, werden für jede Gebäudekomponente geprüft. Dabei wird untersucht, ob die Grenzwerte aus BauO NRW §6 überschritten werden. Wenn eine Überschreitung der Grenzwerte vorliegt, wird eine Abstandsfläche berechnet.
Die Abstandsflächenberechnung selbst findet auf Polygon-Ebene der Multi-Flächen-Geometrie der thematischen CityGML-Begrenzungsflächen statt. Die Abstandsflächen werden je Liniensegment, bei horizontaler Ausdehnung, des äußeren Rings des Polygons berechnet. Alle sich aus einer Teilfläche der thematischen Begrenzungsfläche ergebenden Abstandsflächen werden mit der Erzeugerfläche referenziert, um nachvollziehen zu können, welcher Gebäudeteil für welche Abstandsfläche verantwortlich ist.
Nach vollständiger Untersuchung des Gebäudemodells auf Abstandsflächen wird aus den einzelnen Abstandsflächen der Gebäudeteilflächen eine Gesamtfläche erzeugt und extrudiert, um den Abstandsraum des Gebäudes zu erhalten.

Fazit

Ohne die Bereitstellung der Attribute aus den ADEs für jede Begrenzungsfläche, die dessen Rolle in der Abstandsflächenberechnung festlegt, oder definiert, welche als Referenzkomponente z.B. bzgl. des Maß des Hervortretens dient, war es in dieser Arbeit nicht möglich, den Algorithmus zur Abstandsflächenberechnung zu implementieren. Das liegt daran, dass für die Zuweisung einer bestimmten Rolle oder Relevanz für die Abstandsflächenberechnung häufig der Gesamteindruck und Kontext, in welchen die Gebäudekomponente eingebettet ist, ausschlaggebend ist. Dies wird bspw. in den Formulierungen für die Klassifizierung eines eigenständigen oder als Bestandteil des Dachs angesehenen Dachaufbaus, deutlich. In der Handlungsempfehlung für die BauO NRW wird ein Dachaufbau als eigenständiges Bauelement eingestuft, wenn dieser *„[...] bei wertender Betrachtung [...] als selbständiges Bauteil in Erscheinung tritt."* (“BAUO NRW 2018: HANDLUNGSEMPFEHLUNG AUF DER GRUNDLAGE DER DIENSTBESPRECHUNGEN MIT DEN BAUAUF-SICHTSBEHÖRDEN IM OKTOBER/NOVEMBER 2018”, 2018). Mögliche Kriterien berufen sich auf die *"Gestaltung"* oder *"zusätzliche Auswirkungen"* des Bauteils.

Diese Begriffe und Formulierungen sind nur schwer in mathematische Definitionen zu übersetzen und können von Einzelfall zu Einzelfall verschieden bewertet werden. Die Bereitstellung der Attribute muss manuell geschehen. Dazu könnte eine Software mit graphischer Benutzeroberfläche dienen, in welcher den Gebäudekomponenten manuell die jeweilige Rolle zugewiesen werden könnte.
Um die Überschreitung von Grenzwerten der in BauO NRW §6 geregelten Bedingungen für die Notwendigkeit von Abstandsflächen zu berechnen, werden geometrische Analysen an den Gebäudekomponenten durchgeführt.
Die Berechnung geometrischer Eigenschaften wie Parallelität, Orthogonalität, Länge/Distanz oder Winkel, sowie Schnittpunkten von Geraden, oder topologische Relationen führen durch numerische Rundungsfehler zu falschen Einschätzungen, wenn keine Fehlertoleranzen beim Umgang mit diesen Eigenschaften eingeführt werden. Numerische Ungenauigkeiten in Koordinaten oder Ungenauigkeiten in der geometrischen Modellierung des Gebäudemodells müssen bei der Berechnung von geometrischen Eigenschaften ebenfalls durch Toleranzwerte berücksichtigt werden müssen.
Im Falle von der Berechnung der Schnittpunkte zwischen 3D-Geraden haben numerische oder geometrische Ungenauigkeiten häufig windschiefe Konfigurationen zur Folge. Eine Lösung dafür ist, für die Berechnung der Schnittpunkte die Geraden nach 2D zu transformieren, um die Windschiefe zu eliminieren.
Oft müssen Relationen zwischen planaren Polygon bestimmt werden. In diesem Fall bietet es sich an, eine Transformation in die ”Polygon”-Basis eines der Polygone durchzuführen, sodass die ersten beiden Basisvektoren in der Ebene des Referenz-Polygons liegen, und die Flächennormale den dritte Basisvektor darstellt. Generell können so viele geometrische Fragestellungen von 3D nach 2D vereinfacht werden.

6.2.3 Workflow

Zusammenfassung der Ergebnisse

Der in Kapitel 5 beschriebene Workflow zur Erstellung eines 3D-Lageplans wird in der Safe Software FME implementiert. Aus einem konventionellen 2D-Lageplan im CAD-Format wird ein 3D-Lageplan im CityGML-Format erzeugt, der alle im ursprünglichen Lageplan enthaltenen Objekte um Semantik und 3D-Geometrie erweitert beinhaltet.
Die unstrukturierte Geometrie-Objekte der CAD-Zeichnung werden zunächst in einer GIS-Software in zu CityGML-Klassen korrespondierende Objektarten gruppiert und als Shape-Dateien exportiert.
Für die Verarbeitung zu einem Lageplan in der FME werden die benachbarten Gebäude in LOD2, ein DGM, sowie ein DOM* des Lageplan-Gebiets, die Shape-Dateien der Lageplan-Objekte, sowie das Gebäudemodell des geplanten Bauwerks eingelesen.
Die 3D-Information für die 3D-Geometrie der Objekte des Lageplans wird aus ursprünglichen Textattributen des CAD-Lageplans, dem DGM und dem DOM gewonnen.
In die Prozessierung in der FME wird der Algorithmus der automatischen Abstandsflächenberechnung integriert.

Für das Schreiben des Lageplans in eine Datei im CityGML-Format wird die SiteplanADE benötigt. Das gesamte CityGML-Profil für die Darstellung der Topographie und konzeptionellen Domäne eines Lageplans ist in Abschnitt 5.2.2 abgebildet.

Fazit

Aufgrund der unterschiedlichen Typen und Formate von Daten, die für die Erstellung eines 3D-Lageplans nötig sind, bietet sich für die Prozessierung die Safe Software FME an. Vorausgesetzt sie werden unterstützt, können die verschiedenen Datenformate der Eingangsdaten eingelesen und auf Datenformat-neutrale Objekte in FME-interner Darstellung abgebildet werden. Durch die Datenformat-neutrale Repräsentation wird es möglich, die unterschiedlichen Daten miteinander zu verarbeiten und zu verknüpfen.
Das Schreiben der Daten in ein Format eines individuellen Anwendungsschemas wird über den GML-Writer möglich, in dem der Pfad zum entsprechenden CityGML-Profil gesetzt werden muss.
Durch die Benutzung der FME fallen aufwändige Arbeitsschritte wie das Einlesen und Zusammenführen verschiedener Datenformate, sowie das Schreiben in ein anwendungsspezifisches Ausgangsformat weg. Durch die Vielzahl an Transformationswerkzeugen können die Daten zu großem Teil ohne eigene Programmierung verarbeitet werden. Durch die Python-Einbindung (*PythonCreator* und *PythonCaller*) wird es möglich individuelle Lösungen für Probleme zu finden, die nicht durch standardmäßige Werkzeuge der FME bewerkstelligt werden können.
Zudem ist der einmalig definierte Ablauf der Verarbeitungsschritte der FME reproduzierbar und nicht auf einen speziellen Eingangsdatensatz begrenzt. Werden Format und Typ der Eingangsdaten für die Prozessierung in der FME definiert, kann für jeden Auftrag dieselbe FME-Prozessierungabfolge zur Anwendung kommen und muss nicht jedes Mal erneut Projekt-spezifisch angepasst werden.

7 Ausblick

7.1 Ausblick in Bezug auf den digitalen Lageplan in 3D

Die Erstellung eines Datenmodells für die Bereitstellung eines Lageplans, welches Semantik und Geometrie vereint, und damit für Maschinen interpretierbar macht, ist in den Kontext der Automatisierung und Digitalisierung eingebettet (vgl. Abschnitt 1.1 und Abschnitt 2.2).
Der semantische 3D-Lageplan bietet die Möglichkeit, dass die Prüfung auf formale Einhaltung von enthaltenen Objekten und Modellierungskonventionen, sowie die Gültigkeit des Vorhabens im Kontext des spezifischen Anwendungsfalls, in Zukunft von Maschinen übernommen werden können.
Mit dem Maschinen-interpretierbaren Format des Lageplans lassen sich die Verarbeitungsschritte bzgl. des Lageplans im jeweiligen Anwendungsfall in BPMN und DMN definieren und von *Business Process Engines* ausführen.
Sogenannte *Frameworks* für die Implementierung von automatisierbaren in BPMN definierten Prozess-Abläufen sind z.B. die frei verfügbare *Process Engine* von Mind5 (`https://www.process-engine.io/`) oder *Camunda* (`https://camunda.com/download/`). Andere Workflow-Frameworks, die nicht auf BPMN basieren sind bspw. *Pegasus* (`https://pegasus.isi.edu/`), *Taverna Workflow System* `https://taverna.incubator.apache.org/` oder auch die in der Arbeit verwendete FME von Safe Software (`https://www.safe.com/`).
Die einzelnen Prozesse werden durch den*die Software-Entwickler*in in der entsprechenden Programmiersprache, für die eine Einbindung in die *Business Process Engines* existiert, implementiert und/oder aus bestehenden Prozesskomponenten zusammengefügt.

Durch das XML-basierte Format der SiteplanADE als Anwendungserweiterung des CityGML-Standards werden automatisierbare Prüfungs- und Verarbeitungsschritte aus der XML-Technologie wie XSD-Validierung, Schematron (QuickFix), XSLT und XProc möglich.
Mit XProc kann eine sogenannte *Pipeline* zur Datenverarbeitung in XML definiert werden (Siegel, 2019; Walsh, Milowski & Thompson, 2010). Ein XProc-Prozessor wendet anhand der in der XProc-Datei definierten Prozessierung, ein Reihe verschiedener Verarbeitungsschritte auf die Eingangsdaten an (Siegel, 2019; Walsh et al., 2010). V.a. für die Automatisierung der Validierung eines Lageplan-Instanz-Dokuments gegen XSD- und Schematron-Schema-Dateien könnte XProc in Betracht gezogen werden.

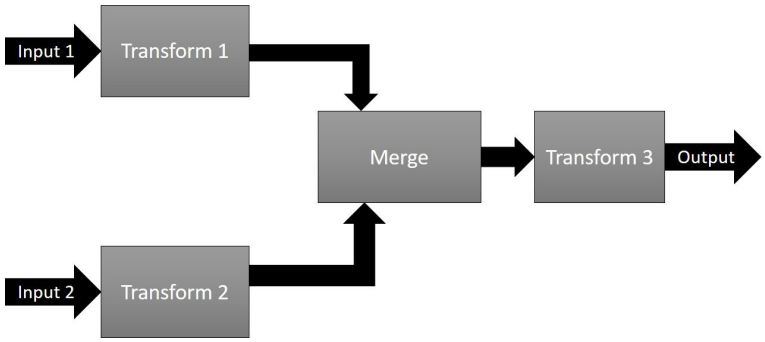


Abbildung 103 Schematische Darstellung einer XProc-Pipeline für die Verarbeitung zweier Eingangsdokumente, und der Ausgabe des Ergebnisdokuments (Siegel, 2019).

Die XProc-Prozessierung zur Validierung des Lageplan-Dokuments könnte als Prozess in den automatisierten Workflow im Kontext eines Anwendungsfalls integriert werden (siehe Abbildung 104). Bevor das XML-Dokument in die restlichen Prozesse eingehen und zusammen mit anderen Eingangsdaten bearbeitet werden kann, muss der Validierungsprozess, welcher auf den jeweiligen Anwendungsfall individualisiert ist, erfolgreich abgeschlossen sein.

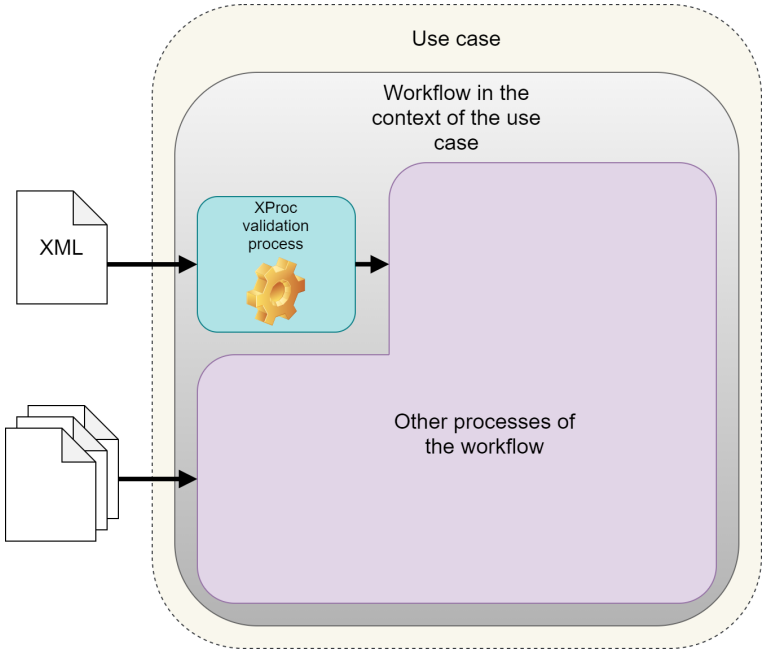


Abbildung 104 In einem Anwendungsfall des Lageplans wird ein Workflow-Diagramm für die Prozessierungsabfolge erstellt. Die Validierung des XML-Dokuments wird in diesen Workflow als Prozess definiert, der mittels XProc die Gültigkeit des Dokuments im Kontext des Anwendungsfalls überprüft, bevor es in den restlichen Teil des Workflows zusammen mit den anderen Eingangsdaten verarbeitet wird.

7.2 Ausblick in Bezug auf die automatisierte Abstandsflächenberechnung

Nach der Beschreibung für Möglichkeiten der Automatisierung anhand eines bereits vorhandenen Lageplan-Instanz-Dokuments in Abschnitt 7.1, wird im Folgenden darauf eingegangen, welche Potenziale der Algorithmus für die automatisierte Abstandsflächenberechnung in der Automatisierung des Workflows bei der Erstellung eines Lageplans bieten könnte.
In Abbildung 105 ist eine schematische Darstellung einer möglichen Automatisierung des Workflows zur Erstellung eines Lageplans abgebildet. Von dem*der Verfasser*in wird nur noch das geplante Gebäudemodell mit Transformationsparametern in das Ziel-Referenzkoordinatensystem, sowie die Gebietsgrenzen des Lageplans vor der Ausführung des Workflows bereitgestellt. Basierend auf den Gebietsgrenzen werden dann automatisiert Anfragen (*Requests*) an entsprechende WFSs (Web Feature Services) gestellt, um Vektordaten bzgl. des Baugebiets (XPlanung) und des Katasters (ALKIS) herunterzuladen. Dasselbe wird für den Bezug von DGM und DOM* durchgeführt. Aus einer 3D-Stadtmodell-Datenbank werden Nachbargebäude, sowie eine definierte Auswahl zusätzlicher Objekte angefordert.
Für die Prozessierung der Daten zu einem Lageplan und der Integration der automatisierten Abstandsflächenberechnung könnte eine FME eingebunden werden (vgl. Unterabschnitt 5.2.2). Diese schreibt nach der entsprechenden Verarbeitung der Daten die Lageplan-Datei und liefert diese zurück in die Workflow-Umgebung.

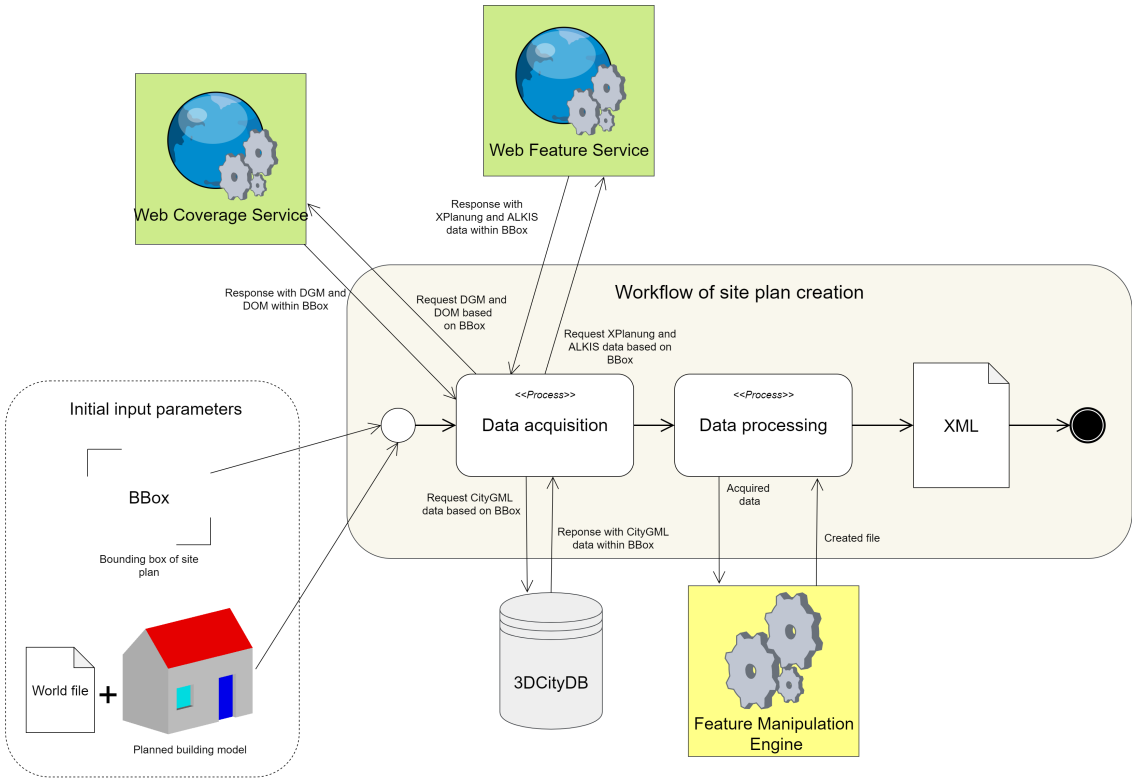


Abbildung 105 Schematische Darstellung der Vollautomatisierten Erstellung eines Lageplans.

7.3 Zusammenfassender Ausblick mit Bezug auf semantische Datenmodelle und Automatisierung

Zusammenfassend lässt sich sagen, dass semantische Datenmodelle einen wichtigen Baustein in der Entwicklung von automatisierten Datenverarbeitungssystemen darstellen. Durch den semantischen Charakter der Datenmodelle wird es möglich, Software zu entwickeln, die Analysen und Berechnungen auf Daten ausführen, welche zuvor nur durch menschliche Interpretation zu bewältigen waren (z.B. die Abstandsflächenberechnung). Diese Algorithmen und Software-Lösungen lassen sich in ausführbare BPMN-Workflow-Umgebungen integrieren, in denen der Mensch nur noch an ausgewählten Punkten mit den Daten interagiert.
Durch die Automatisierung in der Datenverarbeitung lässt sich die Reproduzierbarkeit und Konstanz und damit die Konsistenz der Resultate erhöhen, da die Abfolge der Prozesse, der Datenfluss, und Entscheidungsregeln klar in BPMN-Prozess-Beschreibungen definiert sind und jedes Mal nach demselben Schema abgearbeitet werden.
Die menschliche Komponente konzentriert sich nicht mehr auf die Bearbeitung der Daten selbst, sondern auf die Modellierung von Bearbeitung und Fluss der Daten, sowie die Implementierung der Prozessbeschreibung.

A Gebäude-Profil CityGML 2.0

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
      xmlns:buildingProfile="http://www.citygml.org/buildingProfile" xmlns:core="http://www.opengis.net/citygml/2.0"
      xmlns:gml="http://www.opengis.net/gml" elementFormDefault="qualified" targetNamespace="http://www.citygml.org/
      buildingProfile" version="">
  <import namespace="http://www.opengis.net/citygml/2.0" schemaLocation="http://schemas.opengis.net/citygml/2.0/cityGMLBase
      .xsd"/>
  <import namespace="http://www.opengis.net/citygml/building/2.0" schemaLocation="http://schemas.opengis.net/citygml/
      building/2.0/building.xsd"/>
  <import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
  <!--XML Schema document created by ShapeChange - http://shapechange.net/-->
  <element name="relatedTo" substitutionGroup="core:_GenericApplicationPropertyOfCityObject" type="
      buildingProfile:CityObjectRelationPropertyType"/>
  <element name="CityObjectRelation" substitutionGroup="gml:_GML" type="buildingProfile:CityObjectRelationType"/>
  <complexType name="CityObjectRelationType">
    <complexContent>
      <extension base="gml:AbstractGMLType">
        <sequence>
          <element name="relationType" type="gml:CodeType"/>
          <element name="relatedTo" type="gml:ReferenceType"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="CityObjectRelationPropertyType">
    <sequence minOccurs="0">
      <element ref="buildingProfile:CityObjectRelation"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <element name="GenericThematicSurface" substitutionGroup="bldg:_BoundarySurface" type="
      buildingProfile:GenericThematicSurfaceType"/>
  <complexType name="GenericThematicSurfaceType">
    <complexContent>
      <extension base="bldg:AbstractBoundarySurfaceType">
        <sequence/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="GenericThematicSurfacePropertyType">
    <sequence minOccurs="0">
      <element ref="buildingProfile:GenericThematicSurface"/>
    </sequence>
    <attributeGroup ref="gml:AssociationAttributeGroup"/>
  </complexType>
  <element name="class" substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface" type="gml:CodeType"/>
  <element name="function" substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface" type="gml:CodeType"/>
  <element name="usage" substitutionGroup="bldg:_GenericApplicationPropertyOfBoundarySurface" type="gml:CodeType"/>
</schema>
```

Listing A.1: BuildingProfile.xsd

```
<?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://purl.oclc.org/dsdl/schematron" xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <title>Schematron constraints for schema 'Building'</title>
  <ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>
  <ns prefix="bldg" uri="http://www.opengis.net/citygml/building/2.0"/>
  <!-- Has to be set from 'http://www.opengis.net/gml' to http://www.opengis.net/gml/3.2 for gml 3.2 dictionaries -->
  <ns prefix="gml" uri="http://www.opengis.net/gml"/>
  <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <ns prefix="buildingProfile" uri="http://www.citygml.org/buildingProfile"/>
  <ns prefix="core" uri="http://www.opengis.net/citygml/2.0"/>

  <pattern>
    <rule context="core:CityModel">
      <assert test="count(core:cityObjectMember/*) = 1 and count((core:cityObjectMember/*)[local-name()='Building' and namespace-uri()='http://www.opengis.net/citygml/building/2.0']) = 1">There is only allowed exactly element inside the city model which is of the type 'Building'.</assert>
    </rule>
    <rule context="bldg:Building">
      <assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</assert>
      <assert test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]) &gt; 0">The element must be bounded by thematic surfaces.</assert>
    </rule>
    <rule context="bldg:BuildingInstallation">
      <assert test="bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]">The element must be bounded by thematic surfaces.</assert>
      <assert test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</assert>
    </rule>
    <rule context="bldg:BuildingInstallation/bldg:class">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">Code list shall exist</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a GML 3.2 Dictionary</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">Code list value shall exist</assert>
    </rule>
    <rule context="bldg:BuildingInstallation/bldg:function">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">Code list shall exist</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a GML 3.2 Dictionary</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">Code list value shall exist</assert>
    </rule>
    <rule context="bldg:BuildingInstallation/bldg:usage">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
```

```
      codeListUrl,'#')]))">Code list shall exist</assert>
38    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
39    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
40  </rule>
41  <rule context="bldg:BuildingPart">
42    <assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">Each of the attributes 'class', '
      storeysAboveGround' must be present exactly once.</assert>
43    <assert test="count(bldg:function/text()) = 1">The attribute 'function' must be present exactly once.</assert>
44    <assert test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]) &gt; 0">The
      element must be bounded by thematic surfaces.</assert>
45  </rule>
46  <rule context="bldg:CeilingSurface">
47    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
48  </rule>
49  <rule context="bldg:ClosureSurface">
50    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
51  </rule>
52  <rule context="bldg:FloorSurface">
53    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
54  </rule>
55  <rule context="bldg:InteriorWallSurface">
56    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
57  </rule>
58  <rule context="bldg:OuterCeilingSurface">
59    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
60  </rule>
61  <rule context="bldg:OuterFloorSurface">
62    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
63  </rule>
64  <rule context="bldg:WallSurface">
65    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
66  </rule>
67<rule context="buildingProfile:GenericThematicSurface">
68    <assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1">There must be exactly one element of each attribute 'class', 'function', 'usage'.</
      assert>
69  </rule>
70  <rule context="bldg:Building/bldg:class">
71    <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
72    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
73    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
74    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
```

```
75      </rule>
76      <rule context="bldg:BuildingPart/bldg:class">
77        <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
78        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
79        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
80        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
81      </rule>
82      <rule context="bldg:Building/bldg:function">
83        <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
84        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
85        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
86        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
87      </rule>
88      <rule context="bldg:BuildingPart/bldg:function">
89        <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
90        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
91        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
92        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
93      </rule>
94    <rule context="bldg:OuterCeilingSurface/buildingProfile:class">
95        <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
96        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
97        <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
```

```
98      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
99    </rule>
100   <rule context="bldg:CeilingSurface/buildingProfile:class">
101     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'</assert>
102     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
103     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
        GML 3.2 Dictionary</assert>
104     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
105   </rule>
106   <rule context="bldg:FloorSurface/buildingProfile:class">
107     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'</assert>
108     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
109     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
        GML 3.2 Dictionary</assert>
110     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
111   </rule>
112   <rule context="bldg:ClosureSurface/buildingProfile:class">
113     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'</assert>
114     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
115     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
        GML 3.2 Dictionary</assert>
116     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
117   </rule>
118   <rule context="bldg:OuterFloorSurface/buildingProfile:class">
119     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML31.xml'</assert>
120     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
```

```
121       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
122       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
123     </rule>
124     <rule context="bldg:InteriorWallSurface/buildingProfile:class">
125       <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
126       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
127       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
128       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
129     </rule>
130     <rule context="bldg:WallSurface/buildingProfile:class">
131       <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
132       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
133       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
134       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
135     </rule>
136     <rule context="buildingProfile:GenericThematicSurface/buildingProfile:class">
137       <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
138       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">Code list shall exist</assert>
139       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
140       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
141     </rule>
142     <rule context="bldg:OuterCeilingSurface/buildingProfile:function">
143       <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
```

```
      exampleCodeList_GML31.xml'</assert>
144   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')]))) ">Code list shall exist</assert>
145   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
146   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
147 </rule>
148 <rule context="bldg:CeilingSurface/buildingProfile:function">
149   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
150   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')]))) ">Code list shall exist</assert>
151   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
152   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
153 </rule>
154 <rule context="bldg:FloorSurface/buildingProfile:function">
155   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
156   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')]))) ">Code list shall exist</assert>
157   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
158   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
159 </rule>
160 <rule context="bldg:ClosureSurface/buildingProfile:function">
161   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
162   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')]))) ">Code list shall exist</assert>
163   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
164   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
```

```
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
165   </rule>
166   <rule context="bldg:OuterFloorSurface/buildingProfile:function">
167     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
168     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
169     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
170     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
171   </rule>
172   <rule context="bldg:InteriorWallSurface/buildingProfile:function">
173     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
174     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
175     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
176     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
177   </rule>
178   <rule context="bldg:WallSurface/buildingProfile:function">
179     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
180     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
181     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
182     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
183   </rule>
184   <rule context="buildingProfile:GenericThematicSurface/buildingProfile:function">
185     <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
186     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
187     <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
```

```
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
188   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
189 </rule>
190 <rule context="bldg:OuterCeilingSurface/buildingProfile:usage">
191   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
192   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
193   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
194   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
195 </rule>
196 <rule context="bldg:CeilingSurface/buildingProfile:usage">
197   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
198   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
199   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
200   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
201 </rule>
202 <rule context="bldg:FloorSurface/buildingProfile:usage">
203   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
204   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
205   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
206   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
207 </rule>
208 <rule context="bldg:ClosureSurface/buildingProfile:usage">
209   <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</assert>
210   <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
```

```
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')]))">Code list shall exist</assert>
211 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
212 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
213 </rule>
214 <rule context="bldg:OuterFloorSurface/buildingProfile:usage">
215 <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</assert>
216 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">Code list shall exist</assert>
217 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
218 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
219 </rule>
220 <rule context="bldg:InteriorWallSurface/buildingProfile:usage">
221 <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</assert>
222 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">Code list shall exist</assert>
223 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
224 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
225 </rule>
226 <rule context="bldg:WallSurface/buildingProfile:usage">
227 <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</assert>
228 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">Code list shall exist</assert>
229 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
230 <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
231 </rule>
232 <rule context="buildingProfile:GenericThematicSurface/buildingProfile:usage">
```

```
233       <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</assert>
234       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')]))">Code list shall exist</assert>
235       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
          GML 3.2 Dictionary</assert>
236       <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])">Code list value shall exist</assert>
237     </rule>
238   </pattern>
239 </schema>
```

Listing A.2: schematron_building_profile_citygml2.0.sch

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsl:stylesheet xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:saxon="http://saxon.sf.net/"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:schold="http://www.ascc.net/xml/schematron"
                xmlns:iso="http://purl.oclc.org/dsdl/schematron"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                xmlns:sch="http://purl.oclc.org/dsdl/schematron"
                xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
                xmlns:gml="http://www.opengis.net/gml"
                xmlns:xlink="http://www.w3.org/1999/xlink"
                xmlns:buildingProfile="http://www.citygml.org/buildingProfile"
                xmlns:core="http://www.opengis.net/citygml/2.0"
                version="2.0"><!--Implementers: please note that overriding process-prolog or process-root is
    the preferred method for meta-stylesheets to use where possible. -->
   <xsl:param name="archiveDirParameter"/>
   <xsl:param name="archiveNameParameter"/>
   <xsl:param name="fileNameParameter"/>
   <xsl:param name="fileDirParameter"/>
   <xsl:variable name="document-uri">
      <xsl:value-of select="document-uri(/)"/>
   </xsl:variable>
   <!--PHASES-->
   <!--PROLOG-->
   <xsl:output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
               method="xml"
               omit-xml-declaration="no"
               standalone="yes"
               indent="yes"/>
   <!--XSD TYPES FOR XSLT2-->
   <!--KEYS AND FUNCTIONS-->
   <!--DEFAULT RULES-->
   <!--MODE: SCHEMATRON-SELECT-FULL-PATH-->
   <!--This mode can be used to generate an ugly though full XPath for locators-->
   <xsl:template match="*" mode="schematron-select-full-path">
      <xsl:apply-templates select="." mode="schematron-get-full-path"/>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-->
   <!--This mode can be used to generate an ugly though full XPath for locators-->
   <xsl:template match="*" mode="schematron-get-full-path">
      <xsl:apply-templates select="parent::*" mode="schematron-get-full-path"/>
      <xsl:text>/</xsl:text>
      <xsl:choose>
         <xsl:when test="namespace-uri()=''">
            <xsl:value-of select="name()"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>*:</xsl:text>
            <xsl:value-of select="local-name()"/>
            <xsl:text>[namespace-uri()='</xsl:text>
            <xsl:value-of select="namespace-uri()"/>
            <xsl:text>']</xsl:text>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:variable name="preceding"
                    select="count(preceding-sibling::*[local-name()=local-name(current())
       and namespace-uri() = namespace-uri(current())])"/>
      <xsl:text>[</xsl:text>
      <xsl:value-of select="1+ $preceding"/>
      <xsl:text>]</xsl:text>
   </xsl:template>
   <xsl:template match="@*" mode="schematron-get-full-path">
      <xsl:apply-templates select="parent::*" mode="schematron-get-full-path"/>
      <xsl:text>/</xsl:text>
      <xsl:choose>
         <xsl:when test="namespace-uri()=''">@<xsl:value-of select="name()"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>@*[local-name()='</xsl:text>
            <xsl:value-of select="local-name()"/>
            <xsl:text>' and namespace-uri()='</xsl:text>
            <xsl:value-of select="namespace-uri()"/>
            <xsl:text>']</xsl:text>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-2-->
   <!--This mode can be used to generate prefixed XPath for humans-->
   <xsl:template match="node() | @*" mode="schematron-get-full-path-2">
      <xsl:for-each select="ancestor-or-self::*">
```

```
         <xsl:text>/</xsl:text>
         <xsl:value-of select="name(.)"/>
         <xsl:if test="preceding-sibling::*[name(.)=name(current())]">
            <xsl:text>[</xsl:text>
            <xsl:value-of select="count(preceding-sibling::*[name(.)=name(current())])+1"/>
            <xsl:text>]</xsl:text>
         </xsl:if>
      </xsl:for-each>
      <xsl:if test="not(self::*)">
         <xsl:text/>/@<xsl:value-of select="name(.)"/>
      </xsl:if>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-3-->
   <!--This mode can be used to generate prefixed XPath for humans
  (Top-level element has index)-->
   <xsl:template match="node() | @*" mode="schematron-get-full-path-3">
      <xsl:for-each select="ancestor-or-self::*">
         <xsl:text>/</xsl:text>
         <xsl:value-of select="name(.)"/>
         <xsl:if test="parent::*">
            <xsl:text>[</xsl:text>
            <xsl:value-of select="count(preceding-sibling::*[name(.)=name(current())])+1"/>
            <xsl:text>]</xsl:text>
         </xsl:if>
      </xsl:for-each>
      <xsl:if test="not(self::*)">
         <xsl:text/>/@<xsl:value-of select="name(.)"/>
      </xsl:if>
   </xsl:template>
   <!--MODE: GENERATE-ID-FROM-PATH -->
   <xsl:template match="/" mode="generate-id-from-path"/>
   <xsl:template match="text()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.text-', 1+count(preceding-sibling::text()), '-')"/>
   </xsl:template>
   <xsl:template match="comment()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.comment-', 1+count(preceding-sibling::comment()), '-')"/>
   </xsl:template>
   <xsl:template match="processing-instruction()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.processing-instruction-', 1+count(preceding-sibling::processing-instruction()), '-')"/
      >
   </xsl:template>
   <xsl:template match="@*" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.@', name())"/>
   </xsl:template>
   <xsl:template match="*" mode="generate-id-from-path" priority="-0.5">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:text>.</xsl:text>
      <xsl:value-of select="concat('.',name(),'-',1+count(preceding-sibling::*[name()=name(current())]),'-')"/>
   </xsl:template>
   <!--MODE: GENERATE-ID-2 -->
   <xsl:template match="/" mode="generate-id-2">U</xsl:template>
   <xsl:template match="*" mode="generate-id-2" priority="2">
      <xsl:text>U</xsl:text>
      <xsl:number level="multiple" count="*"/>
   </xsl:template>
   <xsl:template match="node()" mode="generate-id-2">
      <xsl:text>U.</xsl:text>
      <xsl:number level="multiple" count="*"/>
      <xsl:text>n</xsl:text>
      <xsl:number count="node()"/>
   </xsl:template>
   <xsl:template match="@*" mode="generate-id-2">
      <xsl:text>U.</xsl:text>
      <xsl:number level="multiple" count="*"/>
      <xsl:text>_</xsl:text>
      <xsl:value-of select="string-length(local-name(.))"/>
      <xsl:text>_</xsl:text>
      <xsl:value-of select="translate(name(),':','.')"/>
   </xsl:template>
   <!--Strip characters-->
   <xsl:template match="text()" priority="-1"/>
   <!--SCHEMA SETUP-->
   <xsl:template match="/">
      <svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                              title="Schematron constraints for schema 'Building'"
                              schemaVersion="">
         <xsl:comment>
      <xsl:value-of select="$archiveDirParameter"/>
```

```
    <xsl:value-of select="$archiveNameParameter"/>
    <xsl:value-of select="$fileNameParameter"/>
    <xsl:value-of select="$fileDirParameter"/>
        </xsl:comment>
        <svrl:ns-prefix-in-attribute-values uri="http://purl.oclc.org/dsdl/schematron" prefix="sch"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/building/2.0" prefix="bldg"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml" prefix="gml"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.w3.org/1999/xlink" prefix="xlink"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.citygml.org/buildingProfile" prefix="buildingProfile"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/2.0" prefix="core"/>
        <svrl:active-pattern>
           <xsl:attribute name="document">
              <xsl:value-of select="document-uri(/)"/>
           </xsl:attribute>
           <xsl:apply-templates/>
        </svrl:active-pattern>
        <xsl:apply-templates select="/" mode="M7"/>
     </svrl:schematron-output>
  </xsl:template>
  <!--SCHEMATRON PATTERNS-->
  <svrl:text xmlns:svrl="http://purl.oclc.org/dsdl/svrl">Schematron constraints for schema 'Building'</svrl:text>
  <!--PATTERN -->
  <!--RULE -->
  <xsl:template match="core:CityModel" priority="1042" mode="M7">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="core:CityModel"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="count(core:cityObjectMember/*) = 1 and count((core:cityObjectMember/*)[local-name()='Building' and
      namespace-uri()='http://www.opengis.net/citygml/building/2.0']) = 1"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="count(core:cityObjectMember/*) = 1 and count((core:cityObjectMember/*)[local-name()='
     Building' and namespace-uri()='http://www.opengis.net/citygml/building/2.0']) = 1">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>There is only allowed exactly element inside the city model which is of the type 'Building'.</
     svrl:text>
           </svrl:failed-assert>
        </xsl:otherwise>
     </xsl:choose>
     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
  </xsl:template>
  <!--RULE -->
  <xsl:template match="bldg:Building" priority="1041" mode="M7">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:Building"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</svrl:text>
           </svrl:failed-assert>
        </xsl:otherwise>
     </xsl:choose>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]) &gt; 0"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href
     ]) &gt; 0">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
           </svrl:failed-assert>
        </xsl:otherwise>
     </xsl:choose>
     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
  </xsl:template>
  <!--RULE -->
  <xsl:template match="bldg:BuildingInstallation" priority="1040" mode="M7">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                      context="bldg:BuildingInstallation"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]"/>
```

```
240         <xsl:otherwise>
241            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
242                                test="bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]">
243               <xsl:attribute name="location">
244                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
245               </xsl:attribute>
246               <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
247            </svrl:failed-assert>
248         </xsl:otherwise>
249      </xsl:choose>
250      <!--ASSERT -->
251      <xsl:choose>
252         <xsl:when test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage/text()) = 1"
      />
253         <xsl:otherwise>
254            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
255                                test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage
      /text()) = 1">
256               <xsl:attribute name="location">
257                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
258               </xsl:attribute>
259               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
260            </svrl:failed-assert>
261         </xsl:otherwise>
262      </xsl:choose>
263      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
264   </xsl:template>
265   <!--RULE -->
266   <xsl:template match="bldg:BuildingInstallation/bldg:class"
267                 priority="1039"
268                 mode="M7">
269      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
270                       context="bldg:BuildingInstallation/bldg:class"/>
271      <!--ASSERT -->
272      <xsl:choose>
273         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
274         <xsl:otherwise>
275            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
276                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
277               <xsl:attribute name="location">
278                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
279               </xsl:attribute>
280               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
281            </svrl:failed-assert>
282         </xsl:otherwise>
283      </xsl:choose>
284      <!--ASSERT -->
285      <xsl:choose>
286         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')]))) "/>
287         <xsl:otherwise>
288            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
289                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
290               <xsl:attribute name="location">
291                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
292               </xsl:attribute>
293               <svrl:text>Code list shall exist</svrl:text>
294            </svrl:failed-assert>
295         </xsl:otherwise>
296      </xsl:choose>
297      <!--ASSERT -->
298      <xsl:choose>
299         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) "/>
300         <xsl:otherwise>
301            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
302                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
```

```
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
303               <xsl:attribute name="location">
304                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
305               </xsl:attribute>
306               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
307            </svrl:failed-assert>
308         </xsl:otherwise>
309      </xsl:choose>
310      <!--ASSERT -->
311      <xsl:choose>
312         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
313         <xsl:otherwise>
314            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
315                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
316               <xsl:attribute name="location">
317                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
318               </xsl:attribute>
319               <svrl:text>Code list value shall exist</svrl:text>
320            </svrl:failed-assert>
321         </xsl:otherwise>
322      </xsl:choose>
323      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
324   </xsl:template>
325   <!--RULE -->
326   <xsl:template match="bldg:BuildingInstallation/bldg:function"
327                 priority="1038"
328                 mode="M7">
329      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
330                       context="bldg:BuildingInstallation/bldg:function"/>
331      <!--ASSERT -->
332      <xsl:choose>
333         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
334         <xsl:otherwise>
335            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
336                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
337               <xsl:attribute name="location">
338                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
339               </xsl:attribute>
340               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
341            </svrl:failed-assert>
342         </xsl:otherwise>
343      </xsl:choose>
344      <!--ASSERT -->
345      <xsl:choose>
346         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
347         <xsl:otherwise>
348            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
349                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
350               <xsl:attribute name="location">
351                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
352               </xsl:attribute>
353               <svrl:text>Code list shall exist</svrl:text>
354            </svrl:failed-assert>
355         </xsl:otherwise>
356      </xsl:choose>
357      <!--ASSERT -->
358      <xsl:choose>
359         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
```

```
($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))"/>
360         <xsl:otherwise>
361            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
362                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
363               <xsl:attribute name="location">
364                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
365               </xsl:attribute>
366               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
367            </svrl:failed-assert>
368         </xsl:otherwise>
369      </xsl:choose>
370      <!--ASSERT -->
371      <xsl:choose>
372         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
[gml:identifier = $codeListValue])"/>
373         <xsl:otherwise>
374            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
375                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
gml:Definition[gml:identifier = $codeListValue])">
376               <xsl:attribute name="location">
377                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
378               </xsl:attribute>
379               <svrl:text>Code list value shall exist</svrl:text>
380            </svrl:failed-assert>
381         </xsl:otherwise>
382      </xsl:choose>
383      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
384   </xsl:template>
385   <!--RULE -->
386   <xsl:template match="bldg:BuildingInstallation/bldg:usage"
387                 priority="1037"
388                 mode="M7">
389      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
390                       context="bldg:BuildingInstallation/bldg:usage"/>
391      <!--ASSERT -->
392      <xsl:choose>
393         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
394         <xsl:otherwise>
395            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
396                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
397               <xsl:attribute name="location">
398                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
399               </xsl:attribute>
400               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
exampleCodeList_GML31.xml'</svrl:text>
401            </svrl:failed-assert>
402         </xsl:otherwise>
403      </xsl:choose>
404      <!--ASSERT -->
405      <xsl:choose>
406         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')])))"/>
407         <xsl:otherwise>
408            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
409                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
$codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')])))">
410               <xsl:attribute name="location">
411                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
412               </xsl:attribute>
413               <svrl:text>Code list shall exist</svrl:text>
414            </svrl:failed-assert>
```

```
415          </xsl:otherwise>
416       </xsl:choose>
417       <!--ASSERT -->
418       <xsl:choose>
419          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
420          <xsl:otherwise>
421             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
422                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
          substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
          exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
423                <xsl:attribute name="location">
424                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
425                </xsl:attribute>
426                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
427             </svrl:failed-assert>
428          </xsl:otherwise>
429       </xsl:choose>
430       <!--ASSERT -->
431       <xsl:choose>
432          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])"/>
433          <xsl:otherwise>
434             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
435                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
          codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
          gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
          ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
          gml:Definition[gml:identifier = $codeListValue])">
436                <xsl:attribute name="location">
437                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
438                </xsl:attribute>
439                <svrl:text>Code list value shall exist</svrl:text>
440             </svrl:failed-assert>
441          </xsl:otherwise>
442       </xsl:choose>
443       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
444    </xsl:template>
445    <!--RULE -->
446    <xsl:template match="bldg:BuildingPart" priority="1036" mode="M7">
447       <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:BuildingPart"/>
448       <!--ASSERT -->
449       <xsl:choose>
450          <xsl:when test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"/>
451          <xsl:otherwise>
452             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
453                                 test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">
454                <xsl:attribute name="location">
455                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
456                </xsl:attribute>
457                <svrl:text>Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</svrl:text>
458             </svrl:failed-assert>
459          </xsl:otherwise>
460       </xsl:choose>
461       <!--ASSERT -->
462       <xsl:choose>
463          <xsl:when test="count(bldg:function/text()) = 1"/>
464          <xsl:otherwise>
465             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
466                                 test="count(bldg:function/text()) = 1">
467                <xsl:attribute name="location">
468                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
469                </xsl:attribute>
470                <svrl:text>The attribute 'function' must be present exactly once.</svrl:text>
471             </svrl:failed-assert>
472          </xsl:otherwise>
473       </xsl:choose>
474       <!--ASSERT -->
475       <xsl:choose>
476          <xsl:when test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]) &gt; 0"/>
477          <xsl:otherwise>
478             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
```

```
                                test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href
    ]) &gt; 0">
                  <xsl:attribute name="location">
                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
                  </xsl:attribute>
                  <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
               </svrl:failed-assert>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
      </xsl:template>
      <!--RULE -->
      <xsl:template match="bldg:CeilingSurface" priority="1035" mode="M7">
         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:CeilingSurface"/>
         <!--ASSERT -->
         <xsl:choose>
            <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
    buildingProfile:usage/text()) = 1"/>
            <xsl:otherwise>
               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                   test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
    1 and count(buildingProfile:usage/text()) = 1">
                  <xsl:attribute name="location">
                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
                  </xsl:attribute>
                  <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
               </svrl:failed-assert>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
      </xsl:template>
      <!--RULE -->
      <xsl:template match="bldg:ClosureSurface" priority="1034" mode="M7">
         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:ClosureSurface"/>
         <!--ASSERT -->
         <xsl:choose>
            <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
    buildingProfile:usage/text()) = 1"/>
            <xsl:otherwise>
               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                   test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
    1 and count(buildingProfile:usage/text()) = 1">
                  <xsl:attribute name="location">
                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
                  </xsl:attribute>
                  <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
               </svrl:failed-assert>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
      </xsl:template>
      <!--RULE -->
      <xsl:template match="bldg:FloorSurface" priority="1033" mode="M7">
         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:FloorSurface"/>
         <!--ASSERT -->
         <xsl:choose>
            <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
    buildingProfile:usage/text()) = 1"/>
            <xsl:otherwise>
               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                   test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
    1 and count(buildingProfile:usage/text()) = 1">
                  <xsl:attribute name="location">
                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
                  </xsl:attribute>
                  <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
               </svrl:failed-assert>
            </xsl:otherwise>
         </xsl:choose>
         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
      </xsl:template>
      <!--RULE -->
      <xsl:template match="bldg:InteriorWallSurface" priority="1032" mode="M7">
         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                          context="bldg:InteriorWallSurface"/>
         <!--ASSERT -->
         <xsl:choose>
            <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
    buildingProfile:usage/text()) = 1"/>
            <xsl:otherwise>
               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                   test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
```

```
      1 and count(buildingProfile:usage/text()) = 1">
553               <xsl:attribute name="location">
554                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
555               </xsl:attribute>
556               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
557            </svrl:failed-assert>
558         </xsl:otherwise>
559      </xsl:choose>
560      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
561   </xsl:template>
562   <!--RULE -->
563   <xsl:template match="bldg:OuterCeilingSurface" priority="1031" mode="M7">
564      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
565                       context="bldg:OuterCeilingSurface"/>
566      <!--ASSERT -->
567      <xsl:choose>
568         <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1"/>
569         <xsl:otherwise>
570            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
571                                test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
      1 and count(buildingProfile:usage/text()) = 1">
572               <xsl:attribute name="location">
573                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
574               </xsl:attribute>
575               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
576            </svrl:failed-assert>
577         </xsl:otherwise>
578      </xsl:choose>
579      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
580   </xsl:template>
581   <!--RULE -->
582   <xsl:template match="bldg:OuterFloorSurface" priority="1030" mode="M7">
583      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
584                       context="bldg:OuterFloorSurface"/>
585      <!--ASSERT -->
586      <xsl:choose>
587         <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1"/>
588         <xsl:otherwise>
589            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
590                                test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
      1 and count(buildingProfile:usage/text()) = 1">
591               <xsl:attribute name="location">
592                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
593               </xsl:attribute>
594               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
595            </svrl:failed-assert>
596         </xsl:otherwise>
597      </xsl:choose>
598      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
599   </xsl:template>
600   <!--RULE -->
601   <xsl:template match="bldg:WallSurface" priority="1029" mode="M7">
602      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:WallSurface"/>
603      <!--ASSERT -->
604      <xsl:choose>
605         <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1"/>
606         <xsl:otherwise>
607            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
608                                test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
      1 and count(buildingProfile:usage/text()) = 1">
609               <xsl:attribute name="location">
610                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
611               </xsl:attribute>
612               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
613            </svrl:failed-assert>
614         </xsl:otherwise>
615      </xsl:choose>
616      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
617   </xsl:template>
618   <!--RULE -->
619   <xsl:template match="buildingProfile:GenericThematicSurface"
620                 priority="1028"
621                 mode="M7">
622      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
623                       context="buildingProfile:GenericThematicSurface"/>
624      <!--ASSERT -->
625      <xsl:choose>
626         <xsl:when test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and count(
      buildingProfile:usage/text()) = 1"/>
```

```
627         <xsl:otherwise>
628            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
629                                test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) =
    1 and count(buildingProfile:usage/text()) = 1">
630               <xsl:attribute name="location">
631                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
632               </xsl:attribute>
633               <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
634            </svrl:failed-assert>
635         </xsl:otherwise>
636      </xsl:choose>
637      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
638   </xsl:template>
639   <!--RULE -->
640   <xsl:template match="bldg:Building/bldg:class" priority="1027" mode="M7">
641      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
642                       context="bldg:Building/bldg:class"/>
643      <!--ASSERT -->
644      <xsl:choose>
645         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
646         <xsl:otherwise>
647            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
648                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
649               <xsl:attribute name="location">
650                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
651               </xsl:attribute>
652               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</svrl:text>
653            </svrl:failed-assert>
654         </xsl:otherwise>
655      </xsl:choose>
656      <!--ASSERT -->
657      <xsl:choose>
658         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))"/>
659         <xsl:otherwise>
660            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
661                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
    $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">
662               <xsl:attribute name="location">
663                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
664               </xsl:attribute>
665               <svrl:text>Code list shall exist</svrl:text>
666            </svrl:failed-assert>
667         </xsl:otherwise>
668      </xsl:choose>
669      <!--ASSERT -->
670      <xsl:choose>
671         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
672         <xsl:otherwise>
673            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
674                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
675               <xsl:attribute name="location">
676                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
677               </xsl:attribute>
678               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
679            </svrl:failed-assert>
680         </xsl:otherwise>
681      </xsl:choose>
682      <!--ASSERT -->
683      <xsl:choose>
684         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
```

```
      [gml:identifier = $codeListValue])"/>
685         <xsl:otherwise>
686            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
687                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
    codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
    gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
    ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
    gml:Definition[gml:identifier = $codeListValue])">
688               <xsl:attribute name="location">
689                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
690               </xsl:attribute>
691               <svrl:text>Code list value shall exist</svrl:text>
692            </svrl:failed-assert>
693         </xsl:otherwise>
694      </xsl:choose>
695      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
696   </xsl:template>
697   <!--RULE -->
698   <xsl:template match="bldg:BuildingPart/bldg:class" priority="1026" mode="M7">
699      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
700                       context="bldg:BuildingPart/bldg:class"/>
701      <!--ASSERT -->
702      <xsl:choose>
703         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
704         <xsl:otherwise>
705            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
706                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
707               <xsl:attribute name="location">
708                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
709               </xsl:attribute>
710               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</svrl:text>
711            </svrl:failed-assert>
712         </xsl:otherwise>
713      </xsl:choose>
714      <!--ASSERT -->
715      <xsl:choose>
716         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))"/>
717         <xsl:otherwise>
718            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
719                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
    $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">
720               <xsl:attribute name="location">
721                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
722               </xsl:attribute>
723               <svrl:text>Code list shall exist</svrl:text>
724            </svrl:failed-assert>
725         </xsl:otherwise>
726      </xsl:choose>
727      <!--ASSERT -->
728      <xsl:choose>
729         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
730         <xsl:otherwise>
731            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
732                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
733               <xsl:attribute name="location">
734                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
735               </xsl:attribute>
736               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
737            </svrl:failed-assert>
738         </xsl:otherwise>
739      </xsl:choose>
740      <!--ASSERT -->
741      <xsl:choose>
```

```
742          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
             SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
             and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
             codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
             substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
             [gml:identifier = $codeListValue])"/>
743          <xsl:otherwise>
744             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
745                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
             Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
             codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
             gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
             ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
             gml:Definition[gml:identifier = $codeListValue])">
746                <xsl:attribute name="location">
747                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
748                </xsl:attribute>
749                <svrl:text>Code list value shall exist</svrl:text>
750             </svrl:failed-assert>
751          </xsl:otherwise>
752       </xsl:choose>
753       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
754    </xsl:template>
755    <!--RULE -->
756    <xsl:template match="bldg:Building/bldg:function" priority="1025" mode="M7">
757       <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
758                        context="bldg:Building/bldg:function"/>
759       <!--ASSERT -->
760       <xsl:choose>
761          <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
             SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
762          <xsl:otherwise>
763             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
764                                 test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
             SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
765                <xsl:attribute name="location">
766                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
767                </xsl:attribute>
768                <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
             exampleCodeList_GML31.xml'</svrl:text>
769             </svrl:failed-assert>
770          </xsl:otherwise>
771       </xsl:choose>
772       <!--ASSERT -->
773       <xsl:choose>
774          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
             SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
             codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
             codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
             codeListUrl,'#')])))"/>
775          <xsl:otherwise>
776             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
777                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
             Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
             ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
             $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
             codeListUrl,'#')])))">
778                <xsl:attribute name="location">
779                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
780                </xsl:attribute>
781                <svrl:text>Code list shall exist</svrl:text>
782             </svrl:failed-assert>
783          </xsl:otherwise>
784       </xsl:choose>
785       <!--ASSERT -->
786       <xsl:choose>
787          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
             SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
             codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
             ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
             //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
788          <xsl:otherwise>
789             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
790                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
             Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
             ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
             substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
             exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
791                <xsl:attribute name="location">
792                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
793                </xsl:attribute>
794                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
```

```
            </svrl:failed-assert>
         </xsl:otherwise>
      </xsl:choose>
      <!--ASSERT -->
      <xsl:choose>
         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])"/>
         <xsl:otherwise>
            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">
               <xsl:attribute name="location">
                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
               </xsl:attribute>
               <svrl:text>Code list value shall exist</svrl:text>
            </svrl:failed-assert>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
   </xsl:template>
   <!--RULE -->
   <xsl:template match="bldg:BuildingPart/bldg:function" priority="1024" mode="M7">
      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                       context="bldg:BuildingPart/bldg:function"/>
      <!--ASSERT -->
      <xsl:choose>
         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
         <xsl:otherwise>
            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
               <xsl:attribute name="location">
                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
               </xsl:attribute>
               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</svrl:text>
            </svrl:failed-assert>
         </xsl:otherwise>
      </xsl:choose>
      <!--ASSERT -->
      <xsl:choose>
         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))"/>
         <xsl:otherwise>
            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">
               <xsl:attribute name="location">
                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
               </xsl:attribute>
               <svrl:text>Code list shall exist</svrl:text>
            </svrl:failed-assert>
         </xsl:otherwise>
      </xsl:choose>
      <!--ASSERT -->
      <xsl:choose>
         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
         <xsl:otherwise>
            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
```

```
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
849               <xsl:attribute name="location">
850                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
851               </xsl:attribute>
852               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
853            </svrl:failed-assert>
854         </xsl:otherwise>
855      </xsl:choose>
856      <!--ASSERT -->
857      <xsl:choose>
858         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
859         <xsl:otherwise>
860            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
861                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
862               <xsl:attribute name="location">
863                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
864               </xsl:attribute>
865               <svrl:text>Code list value shall exist</svrl:text>
866            </svrl:failed-assert>
867         </xsl:otherwise>
868      </xsl:choose>
869      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
870   </xsl:template>
871   <!--RULE -->
872   <xsl:template match="bldg:OuterCeilingSurface/buildingProfile:class"
873                 priority="1023"
874                 mode="M7">
875      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
876                       context="bldg:OuterCeilingSurface/buildingProfile:class"/>
877      <!--ASSERT -->
878      <xsl:choose>
879         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
880         <xsl:otherwise>
881            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
882                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
883               <xsl:attribute name="location">
884                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
885               </xsl:attribute>
886               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
887            </svrl:failed-assert>
888         </xsl:otherwise>
889      </xsl:choose>
890      <!--ASSERT -->
891      <xsl:choose>
892         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
893         <xsl:otherwise>
894            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
895                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
896               <xsl:attribute name="location">
897                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
898               </xsl:attribute>
899               <svrl:text>Code list shall exist</svrl:text>
900            </svrl:failed-assert>
901         </xsl:otherwise>
902      </xsl:choose>
903      <!--ASSERT -->
904      <xsl:choose>
905         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
```

```
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) "/>
906        <xsl:otherwise>
907           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
908                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
           ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
           substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
           exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
909              <xsl:attribute name="location">
910                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
911              </xsl:attribute>
912              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
913           </svrl:failed-assert>
914        </xsl:otherwise>
915     </xsl:choose>
916     <!--ASSERT -->
917     <xsl:choose>
918        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
           and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
           codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
           substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
           [gml:identifier = $codeListValue])"/>
919        <xsl:otherwise>
920           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
921                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
           codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
           gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
           ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
           gml:Definition[gml:identifier = $codeListValue])">
922              <xsl:attribute name="location">
923                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
924              </xsl:attribute>
925              <svrl:text>Code list value shall exist</svrl:text>
926           </svrl:failed-assert>
927        </xsl:otherwise>
928     </xsl:choose>
929     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
930  </xsl:template>
931  <!--RULE -->
932  <xsl:template match="bldg:CeilingSurface/buildingProfile:class"
933                priority="1022"
934                mode="M7">
935     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
936                      context="bldg:CeilingSurface/buildingProfile:class"/>
937     <!--ASSERT -->
938     <xsl:choose>
939        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
940        <xsl:otherwise>
941           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
942                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
943              <xsl:attribute name="location">
944                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
945              </xsl:attribute>
946              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
           exampleCodeList_GML31.xml'</svrl:text>
947           </svrl:failed-assert>
948        </xsl:otherwise>
949     </xsl:choose>
950     <!--ASSERT -->
951     <xsl:choose>
952        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
           codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
           codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
           codeListUrl,'#')])))"/>
953        <xsl:otherwise>
954           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
955                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
           ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
           $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
           codeListUrl,'#')])))">
956              <xsl:attribute name="location">
957                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
958              </xsl:attribute>
959              <svrl:text>Code list shall exist</svrl:text>
960           </svrl:failed-assert>
961        </xsl:otherwise>
```

```
962         </xsl:choose>
963         <!--ASSERT -->
964         <xsl:choose>
965            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
            ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
            //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
966            <xsl:otherwise>
967               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
968                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
            substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
            exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
969                  <xsl:attribute name="location">
970                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
971                  </xsl:attribute>
972                  <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
973               </svrl:failed-assert>
974            </xsl:otherwise>
975         </xsl:choose>
976         <!--ASSERT -->
977         <xsl:choose>
978            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
            and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
            codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
            substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
            [gml:identifier = $codeListValue])"/>
979            <xsl:otherwise>
980               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
981                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
            codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
            gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
            ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
            gml:Definition[gml:identifier = $codeListValue])">
982                  <xsl:attribute name="location">
983                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
984                  </xsl:attribute>
985                  <svrl:text>Code list value shall exist</svrl:text>
986               </svrl:failed-assert>
987            </xsl:otherwise>
988         </xsl:choose>
989         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
990      </xsl:template>
991      <!--RULE -->
992      <xsl:template match="bldg:FloorSurface/buildingProfile:class"
993                    priority="1021"
994                    mode="M7">
995         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
996                          context="bldg:FloorSurface/buildingProfile:class"/>
997         <!--ASSERT -->
998         <xsl:choose>
999            <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1000            <xsl:otherwise>
1001               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1002                                   test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1003                  <xsl:attribute name="location">
1004                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1005                  </xsl:attribute>
1006                  <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
            exampleCodeList_GML31.xml'</svrl:text>
1007               </svrl:failed-assert>
1008            </xsl:otherwise>
1009         </xsl:choose>
1010         <!--ASSERT -->
1011         <xsl:choose>
1012            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
            codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')])))"/>
1013            <xsl:otherwise>
1014               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1015                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
            $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
```

```
    codeListUrl,'#')]))))">
1016            <xsl:attribute name="location">
1017               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1018            </xsl:attribute>
1019            <svrl:text>Code list shall exist</svrl:text>
1020         </svrl:failed-assert>
1021      </xsl:otherwise>
1022   </xsl:choose>
1023   <!--ASSERT -->
1024   <xsl:choose>
1025      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
       codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
       ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
       //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1026      <xsl:otherwise>
1027         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1028                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
       ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
       substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
       exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
1029            <xsl:attribute name="location">
1030               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1031            </xsl:attribute>
1032            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1033         </svrl:failed-assert>
1034      </xsl:otherwise>
1035   </xsl:choose>
1036   <!--ASSERT -->
1037   <xsl:choose>
1038      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
       and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
       codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
       substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
       [gml:identifier = $codeListValue])"/>
1039      <xsl:otherwise>
1040         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1041                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
       codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
       gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
       ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
       gml:Definition[gml:identifier = $codeListValue])">
1042            <xsl:attribute name="location">
1043               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1044            </xsl:attribute>
1045            <svrl:text>Code list value shall exist</svrl:text>
1046         </svrl:failed-assert>
1047      </xsl:otherwise>
1048   </xsl:choose>
1049   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1050 </xsl:template>
1051 <!--RULE -->
1052 <xsl:template match="bldg:ClosureSurface/buildingProfile:class"
1053               priority="1020"
1054               mode="M7">
1055   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1056                    context="bldg:ClosureSurface/buildingProfile:class"/>
1057   <!--ASSERT -->
1058   <xsl:choose>
1059      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1060      <xsl:otherwise>
1061         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1062                             test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1063            <xsl:attribute name="location">
1064               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1065            </xsl:attribute>
1066            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
       exampleCodeList_GML31.xml'</svrl:text>
1067         </svrl:failed-assert>
1068      </xsl:otherwise>
1069   </xsl:choose>
1070   <!--ASSERT -->
1071   <xsl:choose>
1072      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
       codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
       codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
```

```
codeListUrl,'#')]))))"/>
1073          <xsl:otherwise>
1074             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1075                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
$codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')])))">
1076                <xsl:attribute name="location">
1077                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1078                </xsl:attribute>
1079                <svrl:text>Code list shall exist</svrl:text>
1080             </svrl:failed-assert>
1081          </xsl:otherwise>
1082       </xsl:choose>
1083       <!--ASSERT -->
1084       <xsl:choose>
1085          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
1086          <xsl:otherwise>
1087             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1088                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
1089                <xsl:attribute name="location">
1090                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1091                </xsl:attribute>
1092                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1093             </svrl:failed-assert>
1094          </xsl:otherwise>
1095       </xsl:choose>
1096       <!--ASSERT -->
1097       <xsl:choose>
1098          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
[gml:identifier = $codeListValue])"/>
1099          <xsl:otherwise>
1100             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1101                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
gml:Definition[gml:identifier = $codeListValue])">
1102                <xsl:attribute name="location">
1103                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1104                </xsl:attribute>
1105                <svrl:text>Code list value shall exist</svrl:text>
1106             </svrl:failed-assert>
1107          </xsl:otherwise>
1108       </xsl:choose>
1109       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1110    </xsl:template>
1111    <!--RULE -->
1112    <xsl:template match="bldg:OuterFloorSurface/buildingProfile:class"
1113                  priority="1019"
1114                  mode="M7">
1115       <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1116                        context="bldg:OuterFloorSurface/buildingProfile:class"/>
1117       <!--ASSERT -->
1118       <xsl:choose>
1119          <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1120          <xsl:otherwise>
1121             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1122                                 test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1123                <xsl:attribute name="location">
1124                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1125                </xsl:attribute>
1126                <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
exampleCodeList_GML31.xml'</svrl:text>
1127             </svrl:failed-assert>
1128          </xsl:otherwise>
```

```
1129        </xsl:choose>
1130        <!--ASSERT -->
1131        <xsl:choose>
1132           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
            codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')])))"/>
1133           <xsl:otherwise>
1134              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1135                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
            $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')])))">
1136                 <xsl:attribute name="location">
1137                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1138                 </xsl:attribute>
1139                 <svrl:text>Code list shall exist</svrl:text>
1140              </svrl:failed-assert>
1141           </xsl:otherwise>
1142        </xsl:choose>
1143        <!--ASSERT -->
1144        <xsl:choose>
1145           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
            ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
            //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1146           <xsl:otherwise>
1147              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1148                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
            substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
            exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1149                 <xsl:attribute name="location">
1150                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1151                 </xsl:attribute>
1152                 <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1153              </svrl:failed-assert>
1154           </xsl:otherwise>
1155        </xsl:choose>
1156        <!--ASSERT -->
1157        <xsl:choose>
1158           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
            and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
            codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
            substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
            [gml:identifier = $codeListValue])"/>
1159           <xsl:otherwise>
1160              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1161                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
            codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
            gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
            ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
            gml:Definition[gml:identifier = $codeListValue])">
1162                 <xsl:attribute name="location">
1163                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1164                 </xsl:attribute>
1165                 <svrl:text>Code list value shall exist</svrl:text>
1166              </svrl:failed-assert>
1167           </xsl:otherwise>
1168        </xsl:choose>
1169        <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1170     </xsl:template>
1171     <!--RULE -->
1172     <xsl:template match="bldg:InteriorWallSurface/buildingProfile:class"
1173                   priority="1018"
1174                   mode="M7">
1175        <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1176                         context="bldg:InteriorWallSurface/buildingProfile:class"/>
1177        <!--ASSERT -->
1178        <xsl:choose>
1179           <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1180           <xsl:otherwise>
1181              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1182                                  test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
```

```
1183              <xsl:attribute name="location">
1184                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1185              </xsl:attribute>
1186              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
1187           </svrl:failed-assert>
1188        </xsl:otherwise>
1189     </xsl:choose>
1190     <!--ASSERT -->
1191     <xsl:choose>
1192        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
1193        <xsl:otherwise>
1194           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1195                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
1196              <xsl:attribute name="location">
1197                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1198              </xsl:attribute>
1199              <svrl:text>Code list shall exist</svrl:text>
1200           </svrl:failed-assert>
1201        </xsl:otherwise>
1202     </xsl:choose>
1203     <!--ASSERT -->
1204     <xsl:choose>
1205        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1206        <xsl:otherwise>
1207           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1208                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1209              <xsl:attribute name="location">
1210                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1211              </xsl:attribute>
1212              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1213           </svrl:failed-assert>
1214        </xsl:otherwise>
1215     </xsl:choose>
1216     <!--ASSERT -->
1217     <xsl:choose>
1218        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
1219        <xsl:otherwise>
1220           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1221                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
1222              <xsl:attribute name="location">
1223                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1224              </xsl:attribute>
1225              <svrl:text>Code list value shall exist</svrl:text>
1226           </svrl:failed-assert>
1227        </xsl:otherwise>
1228     </xsl:choose>
1229     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1230  </xsl:template>
1231  <!--RULE -->
1232  <xsl:template match="bldg:WallSurface/buildingProfile:class"
1233                priority="1017"
1234                mode="M7">
1235     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1236                      context="bldg:WallSurface/buildingProfile:class"/>
1237     <!--ASSERT -->
```

```
1238        <xsl:choose>
1239           <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1240           <xsl:otherwise>
1241              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1242                                  test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1243                 <xsl:attribute name="location">
1244                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1245                 </xsl:attribute>
1246                 <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
            exampleCodeList_GML31.xml'</svrl:text>
1247              </svrl:failed-assert>
1248           </xsl:otherwise>
1249        </xsl:choose>
1250        <!--ASSERT -->
1251        <xsl:choose>
1252           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
            codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')]))">
1253           <xsl:otherwise>
1254              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1255                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
            $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')]))">
1256                 <xsl:attribute name="location">
1257                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1258                 </xsl:attribute>
1259                 <svrl:text>Code list shall exist</svrl:text>
1260              </svrl:failed-assert>
1261           </xsl:otherwise>
1262        </xsl:choose>
1263        <!--ASSERT -->
1264        <xsl:choose>
1265           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
            ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
            //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1266           <xsl:otherwise>
1267              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1268                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
            substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
            exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1269                 <xsl:attribute name="location">
1270                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1271                 </xsl:attribute>
1272                 <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1273              </svrl:failed-assert>
1274           </xsl:otherwise>
1275        </xsl:choose>
1276        <!--ASSERT -->
1277        <xsl:choose>
1278           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
            and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
            codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
            substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
            [gml:identifier = $codeListValue])"/>
1279           <xsl:otherwise>
1280              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1281                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
            codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
            gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
            ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
            gml:Definition[gml:identifier = $codeListValue])">
1282                 <xsl:attribute name="location">
1283                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1284                 </xsl:attribute>
1285                 <svrl:text>Code list value shall exist</svrl:text>
1286              </svrl:failed-assert>
1287           </xsl:otherwise>
1288        </xsl:choose>
1289        <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1290     </xsl:template>
```

```
<!--RULE -->
<xsl:template match="buildingProfile:GenericThematicSurface/buildingProfile:class"
              priority="1016"
              mode="M7">
   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                    context="buildingProfile:GenericThematicSurface/buildingProfile:class"/>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
   exampleCodeList_GML31.xml'</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
   codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
   $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
   ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
   //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
   substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
   exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
   and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
   codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
   substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
   [gml:identifier = $codeListValue])"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
   codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
   gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
   ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
   gml:Definition[gml:identifier = $codeListValue])">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
```

```
            </xsl:attribute>
            <svrl:text>Code list value shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
</xsl:template>
<!--RULE -->
<xsl:template match="bldg:OuterCeilingSurface/buildingProfile:function"
              priority="1015"
              mode="M7">
   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                    context="bldg:OuterCeilingSurface/buildingProfile:function"/>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
   exampleCodeList_GML31.xml'</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
   codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
   $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
   ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
   //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
   substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
   exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
   and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
   codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
   substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
   [gml:identifier = $codeListValue])"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
```

```
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
1402              <xsl:attribute name="location">
1403                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1404              </xsl:attribute>
1405              <svrl:text>Code list value shall exist</svrl:text>
1406           </svrl:failed-assert>
1407        </xsl:otherwise>
1408     </xsl:choose>
1409     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1410  </xsl:template>
1411  <!--RULE -->
1412  <xsl:template match="bldg:CeilingSurface/buildingProfile:function"
1413                priority="1014"
1414                mode="M7">
1415     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1416                      context="bldg:CeilingSurface/buildingProfile:function"/>
1417     <!--ASSERT -->
1418     <xsl:choose>
1419        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1420        <xsl:otherwise>
1421           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1422                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1423              <xsl:attribute name="location">
1424                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1425              </xsl:attribute>
1426              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
1427           </svrl:failed-assert>
1428        </xsl:otherwise>
1429     </xsl:choose>
1430     <!--ASSERT -->
1431     <xsl:choose>
1432        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
1433        <xsl:otherwise>
1434           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1435                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
1436              <xsl:attribute name="location">
1437                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1438              </xsl:attribute>
1439              <svrl:text>Code list shall exist</svrl:text>
1440           </svrl:failed-assert>
1441        </xsl:otherwise>
1442     </xsl:choose>
1443     <!--ASSERT -->
1444     <xsl:choose>
1445        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1446        <xsl:otherwise>
1447           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1448                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1449              <xsl:attribute name="location">
1450                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1451              </xsl:attribute>
1452              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1453           </svrl:failed-assert>
1454        </xsl:otherwise>
1455     </xsl:choose>
1456     <!--ASSERT -->
1457     <xsl:choose>
1458        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
```

```
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
1459      <xsl:otherwise>
1460         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1461                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
1462               <xsl:attribute name="location">
1463                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1464               </xsl:attribute>
1465               <svrl:text>Code list value shall exist</svrl:text>
1466            </svrl:failed-assert>
1467         </xsl:otherwise>
1468      </xsl:choose>
1469      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1470   </xsl:template>
1471   <!--RULE -->
1472   <xsl:template match="bldg:FloorSurface/buildingProfile:function"
1473                 priority="1013"
1474                 mode="M7">
1475      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1476                       context="bldg:FloorSurface/buildingProfile:function"/>
1477      <!--ASSERT -->
1478      <xsl:choose>
1479         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1480         <xsl:otherwise>
1481            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1482                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1483               <xsl:attribute name="location">
1484                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1485               </xsl:attribute>
1486               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
1487            </svrl:failed-assert>
1488         </xsl:otherwise>
1489      </xsl:choose>
1490      <!--ASSERT -->
1491      <xsl:choose>
1492         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
1493         <xsl:otherwise>
1494            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1495                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
1496               <xsl:attribute name="location">
1497                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1498               </xsl:attribute>
1499               <svrl:text>Code list shall exist</svrl:text>
1500            </svrl:failed-assert>
1501         </xsl:otherwise>
1502      </xsl:choose>
1503      <!--ASSERT -->
1504      <xsl:choose>
1505         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1506         <xsl:otherwise>
1507            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1508                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1509               <xsl:attribute name="location">
1510                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1511               </xsl:attribute>
1512               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
```

```
1513            </svrl:failed-assert>
1514         </xsl:otherwise>
1515      </xsl:choose>
1516      <!--ASSERT -->
1517      <xsl:choose>
1518         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])"/>
1519         <xsl:otherwise>
1520            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1521                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
          codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
          gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
          ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
          gml:Definition[gml:identifier = $codeListValue])">
1522               <xsl:attribute name="location">
1523                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1524               </xsl:attribute>
1525               <svrl:text>Code list value shall exist</svrl:text>
1526            </svrl:failed-assert>
1527         </xsl:otherwise>
1528      </xsl:choose>
1529      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1530   </xsl:template>
1531   <!--RULE -->
1532   <xsl:template match="bldg:ClosureSurface/buildingProfile:function"
1533                 priority="1012"
1534                 mode="M7">
1535      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1536                       context="bldg:ClosureSurface/buildingProfile:function"/>
1537      <!--ASSERT -->
1538      <xsl:choose>
1539         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1540         <xsl:otherwise>
1541            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1542                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1543               <xsl:attribute name="location">
1544                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1545               </xsl:attribute>
1546               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</svrl:text>
1547            </svrl:failed-assert>
1548         </xsl:otherwise>
1549      </xsl:choose>
1550      <!--ASSERT -->
1551      <xsl:choose>
1552         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))"/>
1553         <xsl:otherwise>
1554            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1555                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
          $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">
1556               <xsl:attribute name="location">
1557                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1558               </xsl:attribute>
1559               <svrl:text>Code list shall exist</svrl:text>
1560            </svrl:failed-assert>
1561         </xsl:otherwise>
1562      </xsl:choose>
1563      <!--ASSERT -->
1564      <xsl:choose>
1565         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1566         <xsl:otherwise>
1567            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1568                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
```

```
($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1569            <xsl:attribute name="location">
1570               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1571            </xsl:attribute>
1572            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1573         </svrl:failed-assert>
1574      </xsl:otherwise>
1575   </xsl:choose>
1576   <!--ASSERT -->
1577   <xsl:choose>
1578      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
[gml:identifier = $codeListValue])"/>
1579      <xsl:otherwise>
1580         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1581                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
gml:Definition[gml:identifier = $codeListValue])">
1582            <xsl:attribute name="location">
1583               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1584            </xsl:attribute>
1585            <svrl:text>Code list value shall exist</svrl:text>
1586         </svrl:failed-assert>
1587      </xsl:otherwise>
1588   </xsl:choose>
1589   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1590</xsl:template>
1591<!--RULE -->
1592<xsl:template match="bldg:OuterFloorSurface/buildingProfile:function"
1593               priority="1011"
1594               mode="M7">
1595   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1596                    context="bldg:OuterFloorSurface/buildingProfile:function"/>
1597   <!--ASSERT -->
1598   <xsl:choose>
1599      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1600      <xsl:otherwise>
1601         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1602                             test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1603            <xsl:attribute name="location">
1604               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1605            </xsl:attribute>
1606            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
exampleCodeList_GML31.xml'</svrl:text>
1607         </svrl:failed-assert>
1608      </xsl:otherwise>
1609   </xsl:choose>
1610   <!--ASSERT -->
1611   <xsl:choose>
1612      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')])))"/>
1613      <xsl:otherwise>
1614         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1615                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
$codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')])))">
1616            <xsl:attribute name="location">
1617               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1618            </xsl:attribute>
1619            <svrl:text>Code list shall exist</svrl:text>
1620         </svrl:failed-assert>
1621      </xsl:otherwise>
1622   </xsl:choose>
1623   <!--ASSERT -->
1624   <xsl:choose>
1625      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
```

```
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) "/>
1626          <xsl:otherwise>
1627             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1628                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) ">
1629                <xsl:attribute name="location">
1630                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1631                </xsl:attribute>
1632                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1633             </svrl:failed-assert>
1634          </xsl:otherwise>
1635       </xsl:choose>
1636       <!--ASSERT -->
1637       <xsl:choose>
1638          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies (not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
1639          <xsl:otherwise>
1640             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1641                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies (not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
1642                <xsl:attribute name="location">
1643                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1644                </xsl:attribute>
1645                <svrl:text>Code list value shall exist</svrl:text>
1646             </svrl:failed-assert>
1647          </xsl:otherwise>
1648       </xsl:choose>
1649       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1650    </xsl:template>
1651    <!--RULE -->
1652    <xsl:template match="bldg:InteriorWallSurface/buildingProfile:function"
1653                  priority="1010"
1654                  mode="M7">
1655       <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1656                        context="bldg:InteriorWallSurface/buildingProfile:function"/>
1657       <!--ASSERT -->
1658       <xsl:choose>
1659          <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1660          <xsl:otherwise>
1661             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1662                                 test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1663                <xsl:attribute name="location">
1664                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1665                </xsl:attribute>
1666                <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML31.xml'</svrl:text>
1667             </svrl:failed-assert>
1668          </xsl:otherwise>
1669       </xsl:choose>
1670       <!--ASSERT -->
1671       <xsl:choose>
1672          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
1673          <xsl:otherwise>
1674             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1675                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
1676                <xsl:attribute name="location">
1677                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1678                </xsl:attribute>
1679                <svrl:text>Code list shall exist</svrl:text>
```

```
1680            </svrl:failed-assert>
1681         </xsl:otherwise>
1682      </xsl:choose>
1683      <!--ASSERT -->
1684      <xsl:choose>
1685         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1686         <xsl:otherwise>
1687            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1688                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
          substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
          exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1689               <xsl:attribute name="location">
1690                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1691               </xsl:attribute>
1692               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1693            </svrl:failed-assert>
1694         </xsl:otherwise>
1695      </xsl:choose>
1696      <!--ASSERT -->
1697      <xsl:choose>
1698         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])"/>
1699         <xsl:otherwise>
1700            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1701                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
          codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
          gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
          ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
          gml:Definition[gml:identifier = $codeListValue])">
1702               <xsl:attribute name="location">
1703                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1704               </xsl:attribute>
1705               <svrl:text>Code list value shall exist</svrl:text>
1706            </svrl:failed-assert>
1707         </xsl:otherwise>
1708      </xsl:choose>
1709      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1710   </xsl:template>
1711   <!--RULE -->
1712   <xsl:template match="bldg:WallSurface/buildingProfile:function"
1713                 priority="1009"
1714                 mode="M7">
1715      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1716                       context="bldg:WallSurface/buildingProfile:function"/>
1717      <!--ASSERT -->
1718      <xsl:choose>
1719         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1720         <xsl:otherwise>
1721            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1722                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1723               <xsl:attribute name="location">
1724                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1725               </xsl:attribute>
1726               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
          exampleCodeList_GML31.xml'</svrl:text>
1727            </svrl:failed-assert>
1728         </xsl:otherwise>
1729      </xsl:choose>
1730      <!--ASSERT -->
1731      <xsl:choose>
1732         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))"/>
1733         <xsl:otherwise>
1734            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1735                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
```

```
($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
$codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
codeListUrl,'#')]))))">
1736            <xsl:attribute name="location">
1737               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1738            </xsl:attribute>
1739            <svrl:text>Code list shall exist</svrl:text>
1740         </svrl:failed-assert>
1741      </xsl:otherwise>
1742   </xsl:choose>
1743   <!--ASSERT -->
1744   <xsl:choose>
1745      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
   ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
   //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1746      <xsl:otherwise>
1747         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1748                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
   substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
   exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1749            <xsl:attribute name="location">
1750               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1751            </xsl:attribute>
1752            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1753         </svrl:failed-assert>
1754      </xsl:otherwise>
1755   </xsl:choose>
1756   <!--ASSERT -->
1757   <xsl:choose>
1758      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
   and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
   codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
   substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
   [gml:identifier = $codeListValue])"/>
1759      <xsl:otherwise>
1760         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1761                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
   codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
   gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
   ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
   gml:Definition[gml:identifier = $codeListValue])">
1762            <xsl:attribute name="location">
1763               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1764            </xsl:attribute>
1765            <svrl:text>Code list value shall exist</svrl:text>
1766         </svrl:failed-assert>
1767      </xsl:otherwise>
1768   </xsl:choose>
1769   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1770 </xsl:template>
1771 <!--RULE -->
1772 <xsl:template match="buildingProfile:GenericThematicSurface/buildingProfile:function"
1773               priority="1008"
1774               mode="M7">
1775    <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1776                     context="buildingProfile:GenericThematicSurface/buildingProfile:function"/>
1777    <!--ASSERT -->
1778    <xsl:choose>
1779       <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1780       <xsl:otherwise>
1781          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1782                              test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1783             <xsl:attribute name="location">
1784                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1785             </xsl:attribute>
1786             <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
   exampleCodeList_GML31.xml'</svrl:text>
1787          </svrl:failed-assert>
1788       </xsl:otherwise>
1789    </xsl:choose>
1790    <!--ASSERT -->
1791    <xsl:choose>
1792       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
```

```
codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))"/>
1793        <xsl:otherwise>
1794           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1795                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">
1796              <xsl:attribute name="location">
1797                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1798              </xsl:attribute>
1799              <svrl:text>Code list shall exist</svrl:text>
1800           </svrl:failed-assert>
1801        </xsl:otherwise>
1802     </xsl:choose>
1803     <!--ASSERT -->
1804     <xsl:choose>
1805        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1806        <xsl:otherwise>
1807           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1808                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1809              <xsl:attribute name="location">
1810                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1811              </xsl:attribute>
1812              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1813           </svrl:failed-assert>
1814        </xsl:otherwise>
1815     </xsl:choose>
1816     <!--ASSERT -->
1817     <xsl:choose>
1818        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])"/>
1819        <xsl:otherwise>
1820           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1821                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">
1822              <xsl:attribute name="location">
1823                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1824              </xsl:attribute>
1825              <svrl:text>Code list value shall exist</svrl:text>
1826           </svrl:failed-assert>
1827        </xsl:otherwise>
1828     </xsl:choose>
1829     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1830  </xsl:template>
1831  <!--RULE -->
1832  <xsl:template match="bldg:OuterCeilingSurface/buildingProfile:usage"
1833                priority="1007"
1834                mode="M7">
1835     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1836                      context="bldg:OuterCeilingSurface/buildingProfile:usage"/>
1837     <!--ASSERT -->
1838     <xsl:choose>
1839        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1840        <xsl:otherwise>
1841           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1842                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
1843              <xsl:attribute name="location">
1844                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1845              </xsl:attribute>
1846              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</svrl:text>
```

```
1847            </svrl:failed-assert>
1848         </xsl:otherwise>
1849      </xsl:choose>
1850      <!--ASSERT -->
1851      <xsl:choose>
1852         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))"/>
1853         <xsl:otherwise>
1854            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1855                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
          $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')])))">
1856               <xsl:attribute name="location">
1857                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1858               </xsl:attribute>
1859               <svrl:text>Code list shall exist</svrl:text>
1860            </svrl:failed-assert>
1861         </xsl:otherwise>
1862      </xsl:choose>
1863      <!--ASSERT -->
1864      <xsl:choose>
1865         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
1866         <xsl:otherwise>
1867            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1868                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
          substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
          exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
1869               <xsl:attribute name="location">
1870                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1871               </xsl:attribute>
1872               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
1873            </svrl:failed-assert>
1874         </xsl:otherwise>
1875      </xsl:choose>
1876      <!--ASSERT -->
1877      <xsl:choose>
1878         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])"/>
1879         <xsl:otherwise>
1880            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1881                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
          codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
          gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
          ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
          gml:Definition[gml:identifier = $codeListValue])">
1882               <xsl:attribute name="location">
1883                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
1884               </xsl:attribute>
1885               <svrl:text>Code list value shall exist</svrl:text>
1886            </svrl:failed-assert>
1887         </xsl:otherwise>
1888      </xsl:choose>
1889      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
1890   </xsl:template>
1891   <!--RULE -->
1892   <xsl:template match="bldg:CeilingSurface/buildingProfile:usage"
1893                 priority="1006"
1894                 mode="M7">
1895      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
1896                       context="bldg:CeilingSurface/buildingProfile:usage"/>
1897      <!--ASSERT -->
1898      <xsl:choose>
1899         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
1900         <xsl:otherwise>
1901            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
```

```
                              test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#'))]))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list value shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
</xsl:template>
<!--RULE -->
<xsl:template match="bldg:FloorSurface/buildingProfile:usage"
              priority="1005"
              mode="M7">
   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
```

```
                    context="bldg:FloorSurface/buildingProfile:usage"/>
<!--ASSERT -->
<xsl:choose>
   <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
   <xsl:otherwise>
      <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                          test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
         <xsl:attribute name="location">
            <xsl:apply-templates select="." mode="schematron-select-full-path"/>
         </xsl:attribute>
         <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml'</svrl:text>
      </svrl:failed-assert>
   </xsl:otherwise>
</xsl:choose>
<!--ASSERT -->
<xsl:choose>
   <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))"/>
   <xsl:otherwise>
      <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                          test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($codeListUrl,'#')])))">
         <xsl:attribute name="location">
            <xsl:apply-templates select="." mode="schematron-select-full-path"/>
         </xsl:attribute>
         <svrl:text>Code list shall exist</svrl:text>
      </svrl:failed-assert>
   </xsl:otherwise>
</xsl:choose>
<!--ASSERT -->
<xsl:choose>
   <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
   <xsl:otherwise>
      <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                          test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
         <xsl:attribute name="location">
            <xsl:apply-templates select="." mode="schematron-select-full-path"/>
         </xsl:attribute>
         <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
      </svrl:failed-assert>
   </xsl:otherwise>
</xsl:choose>
<!--ASSERT -->
<xsl:choose>
   <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])"/>
   <xsl:otherwise>
      <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                          test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition[gml:identifier = $codeListValue])">
         <xsl:attribute name="location">
            <xsl:apply-templates select="." mode="schematron-select-full-path"/>
         </xsl:attribute>
         <svrl:text>Code list value shall exist</svrl:text>
      </svrl:failed-assert>
   </xsl:otherwise>
</xsl:choose>
```

```
2009        <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2010     </xsl:template>
2011     <!--RULE -->
2012     <xsl:template match="bldg:ClosureSurface/buildingProfile:usage"
2013                   priority="1004"
2014                   mode="M7">
2015        <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2016                         context="bldg:ClosureSurface/buildingProfile:usage"/>
2017        <!--ASSERT -->
2018        <xsl:choose>
2019           <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
2020           <xsl:otherwise>
2021              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2022                                  test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
2023                 <xsl:attribute name="location">
2024                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2025                 </xsl:attribute>
2026                 <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
           exampleCodeList_GML31.xml'</svrl:text>
2027              </svrl:failed-assert>
2028           </xsl:otherwise>
2029        </xsl:choose>
2030        <!--ASSERT -->
2031        <xsl:choose>
2032           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
           codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
           codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
           codeListUrl,'#')])))"/>
2033           <xsl:otherwise>
2034              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2035                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
           ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
           $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
           codeListUrl,'#')])))">
2036                 <xsl:attribute name="location">
2037                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2038                 </xsl:attribute>
2039                 <svrl:text>Code list shall exist</svrl:text>
2040              </svrl:failed-assert>
2041           </xsl:otherwise>
2042        </xsl:choose>
2043        <!--ASSERT -->
2044        <xsl:choose>
2045           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
           codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
           ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
           //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
2046           <xsl:otherwise>
2047              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2048                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
           ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
           substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
           exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
2049                 <xsl:attribute name="location">
2050                    <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2051                 </xsl:attribute>
2052                 <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
2053              </svrl:failed-assert>
2054           </xsl:otherwise>
2055        </xsl:choose>
2056        <!--ASSERT -->
2057        <xsl:choose>
2058           <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
           SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
           and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
           codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
           substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
           [gml:identifier = $codeListValue])"/>
2059           <xsl:otherwise>
2060              <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2061                                  test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
           Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
           codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
           gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
           ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
           gml:Definition[gml:identifier = $codeListValue])">
```

```
2062              <xsl:attribute name="location">
2063                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2064              </xsl:attribute>
2065              <svrl:text>Code list value shall exist</svrl:text>
2066           </svrl:failed-assert>
2067        </xsl:otherwise>
2068     </xsl:choose>
2069     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2070  </xsl:template>
2071  <!--RULE -->
2072  <xsl:template match="bldg:OuterFloorSurface/buildingProfile:usage"
2073                priority="1003"
2074                mode="M7">
2075     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2076                      context="bldg:OuterFloorSurface/buildingProfile:usage"/>
2077     <!--ASSERT -->
2078     <xsl:choose>
2079        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
2080        <xsl:otherwise>
2081           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2082                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
2083              <xsl:attribute name="location">
2084                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2085              </xsl:attribute>
2086              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
     exampleCodeList_GML31.xml'</svrl:text>
2087           </svrl:failed-assert>
2088        </xsl:otherwise>
2089     </xsl:choose>
2090     <!--ASSERT -->
2091     <xsl:choose>
2092        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
     codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
     codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
     codeListUrl,'#')])))"/>
2093        <xsl:otherwise>
2094           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2095                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
     Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
     ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
     $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
     codeListUrl,'#')])))">
2096              <xsl:attribute name="location">
2097                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2098              </xsl:attribute>
2099              <svrl:text>Code list shall exist</svrl:text>
2100           </svrl:failed-assert>
2101        </xsl:otherwise>
2102     </xsl:choose>
2103     <!--ASSERT -->
2104     <xsl:choose>
2105        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
     codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
     ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
     //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
2106        <xsl:otherwise>
2107           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2108                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
     Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
     ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
     substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
     exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
2109              <xsl:attribute name="location">
2110                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2111              </xsl:attribute>
2112              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
2113           </svrl:failed-assert>
2114        </xsl:otherwise>
2115     </xsl:choose>
2116     <!--ASSERT -->
2117     <xsl:choose>
2118        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
     and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
     codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
     substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
     [gml:identifier = $codeListValue])"/>
2119        <xsl:otherwise>
```

```
2120            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2121                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
    codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
    gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
    ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
    gml:Definition[gml:identifier = $codeListValue])">
2122               <xsl:attribute name="location">
2123                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2124               </xsl:attribute>
2125               <svrl:text>Code list value shall exist</svrl:text>
2126            </svrl:failed-assert>
2127         </xsl:otherwise>
2128      </xsl:choose>
2129      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2130   </xsl:template>
2131   <!--RULE -->
2132   <xsl:template match="bldg:InteriorWallSurface/buildingProfile:usage"
2133                 priority="1002"
2134                 mode="M7">
2135      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2136                       context="bldg:InteriorWallSurface/buildingProfile:usage"/>
2137      <!--ASSERT -->
2138      <xsl:choose>
2139         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
2140         <xsl:otherwise>
2141            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2142                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
2143               <xsl:attribute name="location">
2144                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2145               </xsl:attribute>
2146               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML31.xml'</svrl:text>
2147            </svrl:failed-assert>
2148         </xsl:otherwise>
2149      </xsl:choose>
2150      <!--ASSERT -->
2151      <xsl:choose>
2152         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))"/>
2153         <xsl:otherwise>
2154            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2155                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
    $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">
2156               <xsl:attribute name="location">
2157                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2158               </xsl:attribute>
2159               <svrl:text>Code list shall exist</svrl:text>
2160            </svrl:failed-assert>
2161         </xsl:otherwise>
2162      </xsl:choose>
2163      <!--ASSERT -->
2164      <xsl:choose>
2165         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
2166         <xsl:otherwise>
2167            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2168                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
2169               <xsl:attribute name="location">
2170                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2171               </xsl:attribute>
2172               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
2173            </svrl:failed-assert>
2174         </xsl:otherwise>
2175      </xsl:choose>
2176      <!--ASSERT -->
2177      <xsl:choose>
```

```
2178          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
              SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
              and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
              codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
              substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
              [gml:identifier = $codeListValue])"/>
2179          <xsl:otherwise>
2180             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2181                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
              Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
              codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
              gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#'))
              ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
              gml:Definition[gml:identifier = $codeListValue])">
2182             <xsl:attribute name="location">
2183                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2184             </xsl:attribute>
2185             <svrl:text>Code list value shall exist</svrl:text>
2186          </svrl:failed-assert>
2187       </xsl:otherwise>
2188    </xsl:choose>
2189    <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2190 </xsl:template>
2191 <!--RULE -->
2192 <xsl:template match="bldg:WallSurface/buildingProfile:usage"
2193               priority="1001"
2194               mode="M7">
2195    <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2196                     context="bldg:WallSurface/buildingProfile:usage"/>
2197    <!--ASSERT -->
2198    <xsl:choose>
2199       <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
              SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
2200       <xsl:otherwise>
2201          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2202                              test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
              SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
2203             <xsl:attribute name="location">
2204                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2205             </xsl:attribute>
2206             <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
              exampleCodeList_GML31.xml'</svrl:text>
2207          </svrl:failed-assert>
2208       </xsl:otherwise>
2209    </xsl:choose>
2210    <!--ASSERT -->
2211    <xsl:choose>
2212       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
              SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
              codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
              codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
              codeListUrl,'#')])))"/>
2213       <xsl:otherwise>
2214          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2215                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
              Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
              ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
              $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
              codeListUrl,'#')])))">
2216             <xsl:attribute name="location">
2217                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2218             </xsl:attribute>
2219             <svrl:text>Code list shall exist</svrl:text>
2220          </svrl:failed-assert>
2221       </xsl:otherwise>
2222    </xsl:choose>
2223    <!--ASSERT -->
2224    <xsl:choose>
2225       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
              SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
              codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
              ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
              //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
2226       <xsl:otherwise>
2227          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2228                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
              Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
              ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
              substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
              exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
2229             <xsl:attribute name="location">
2230                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
```

```
2231              </xsl:attribute>
2232              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
2233           </svrl:failed-assert>
2234        </xsl:otherwise>
2235     </xsl:choose>
2236     <!--ASSERT -->
2237     <xsl:choose>
2238        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
         and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
         codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
         substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
         [gml:identifier = $codeListValue])"/>
2239        <xsl:otherwise>
2240           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2241                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
         Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
         codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
         gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
         ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
         gml:Definition[gml:identifier = $codeListValue])">
2242              <xsl:attribute name="location">
2243                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2244              </xsl:attribute>
2245              <svrl:text>Code list value shall exist</svrl:text>
2246           </svrl:failed-assert>
2247        </xsl:otherwise>
2248     </xsl:choose>
2249     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2250  </xsl:template>
2251  <!--RULE -->
2252  <xsl:template match="buildingProfile:GenericThematicSurface/buildingProfile:usage"
2253                priority="1000"
2254                mode="M7">
2255     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2256                      context="buildingProfile:GenericThematicSurface/buildingProfile:usage"/>
2257     <!--ASSERT -->
2258     <xsl:choose>
2259        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML31.xml'"/>
2260        <xsl:otherwise>
2261           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2262                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML31.xml'">
2263              <xsl:attribute name="location">
2264                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2265              </xsl:attribute>
2266              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
         exampleCodeList_GML31.xml'</svrl:text>
2267           </svrl:failed-assert>
2268        </xsl:otherwise>
2269     </xsl:choose>
2270     <!--ASSERT -->
2271     <xsl:choose>
2272        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
         codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
         codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
         codeListUrl,'#')])))"/>
2273        <xsl:otherwise>
2274           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2275                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
         Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
         ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
         $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
         codeListUrl,'#')])))">
2276              <xsl:attribute name="location">
2277                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2278              </xsl:attribute>
2279              <svrl:text>Code list shall exist</svrl:text>
2280           </svrl:failed-assert>
2281        </xsl:otherwise>
2282     </xsl:choose>
2283     <!--ASSERT -->
2284     <xsl:choose>
2285        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
         codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
         ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
         //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
2286        <xsl:otherwise>
2287           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
```

```
2288                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
2289            <xsl:attribute name="location">
2290               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2291            </xsl:attribute>
2292            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
2293         </svrl:failed-assert>
2294      </xsl:otherwise>
2295   </xsl:choose>
2296   <!--ASSERT -->
2297   <xsl:choose>
2298      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])"/>
2299      <xsl:otherwise>
2300         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
2301                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($
    codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
    gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
    ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
    gml:Definition[gml:identifier = $codeListValue])">
2302            <xsl:attribute name="location">
2303               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
2304            </xsl:attribute>
2305            <svrl:text>Code list value shall exist</svrl:text>
2306         </svrl:failed-assert>
2307      </xsl:otherwise>
2308   </xsl:choose>
2309   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2310 </xsl:template>
2311 <xsl:template match="text()" priority="-1" mode="M7"/>
2312 <xsl:template match="@*|node()" priority="-2" mode="M7">
2313    <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M7"/>
2314 </xsl:template>
2315 </xsl:stylesheet>
```

Listing A.3: validationScript.xsl

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                        xmlns:saxon="http://saxon.sf.net/"
                        xmlns:schold="http://www.ascc.net/xml/schematron"
                        xmlns:iso="http://purl.oclc.org/dsdl/schematron"
                        xmlns:xhtml="http://www.w3.org/1999/xhtml"
                        xmlns:sch="http://purl.oclc.org/dsdl/schematron"
                        xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
                        xmlns:gml="http://www.opengis.net/gml"
                        xmlns:xlink="http://www.w3.org/1999/xlink"
                        xmlns:buildingProfile="http://www.citygml.org/buildingProfile"
                        xmlns:core="http://www.opengis.net/citygml/2.0"
                        title="Schematron constraints for schema 'Building'"
                        schemaVersion="">
  <svrl:ns-prefix-in-attribute-values uri="http://purl.oclc.org/dsdl/schematron" prefix="sch"/>
  <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/building/2.0" prefix="bldg"/>
  <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml" prefix="gml"/>
  <svrl:ns-prefix-in-attribute-values uri="http://www.w3.org/1999/xlink" prefix="xlink"/>
  <svrl:ns-prefix-in-attribute-values uri="http://www.citygml.org/buildingProfile" prefix="buildingProfile"/>
  <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/2.0" prefix="core"/>
  <svrl:active-pattern document="file:/C:/Users/felix/Documents/SaxonXLSTProcessor/./data/example.gml"/>
  <svrl:fired-rule context="core:CityModel"/>
  <svrl:failed-assert test="count(core:cityObjectMember/*) = 1 and count((core:cityObjectMember/*)[local-name()='Building'
      and namespace-uri()='http://www.opengis.net/citygml/building/2.0']) = 1"
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]">
      <svrl:text>There is only allowed exactly element inside the city model which is of the type 'Building'.</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="bldg:Building"/>
   <svrl:failed-assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]">
      <svrl:text>Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</svrl:text>
   </svrl:failed-assert>
   <svrl:failed-assert test="count(bldg:boundedBy/* | //*[concat('#',@gml:id)=current()/bldg:boundedBy/@xlink:href]) &gt; 0
      "
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]">
      <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="bldg:Building"/>
   <svrl:failed-assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]">
      <svrl:text>Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="bldg:Building/bldg:class"/>
   <svrl:fired-rule context="bldg:WallSurface"/>
   <svrl:fired-rule context="bldg:WallSurface/buildingProfile:class"/>
   <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:class[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
      <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.
      xml'</svrl:text>
   </svrl:failed-assert>
   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
      $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"
                      location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:class[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
      <svrl:text>Code list value shall exist</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="bldg:WallSurface/buildingProfile:function"/>
   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
      $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
```

```
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"
56                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:function[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
57      <svrl:text>Code list value shall exist</svrl:text>
58    </svrl:failed-assert>
59    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:usage"/>
60    <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml'"
61                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:usage[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
62      <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML31.
      xml'</svrl:text>
63    </svrl:failed-assert>
64    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
      $codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after(
      $codeListUrl,'#')])))"
65                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:usage[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
66      <svrl:text>Code list shall exist</svrl:text>
67    </svrl:failed-assert>
68    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
      $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
69                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:usage[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
70      <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
71    </svrl:failed-assert>
72    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
      $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"
73                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:usage[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
74      <svrl:text>Code list value shall exist</svrl:text>
75    </svrl:failed-assert>
76    <svrl:fired-rule context="bldg:WallSurface"/>
77    <svrl:failed-assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and
      count(buildingProfile:usage/text()) = 1"
78                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][3]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]">
79      <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
80    </svrl:failed-assert>
81    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:class"/>
82    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:class"/>
83    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:function"/>
84    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML31.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
      $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"
85                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][3]/*:WallSurface[
      namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]/*:function[namespace-uri()='http://www.citygml.org/
      buildingProfile'][1]">
86      <svrl:text>Code list value shall exist</svrl:text>
87    </svrl:failed-assert>
```

```
88    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:usage"/>
89    <svrl:fired-rule context="bldg:WallSurface"/>
90    <svrl:failed-assert test="count(buildingProfile:class/text()) = 1 and count(buildingProfile:function/text()) = 1 and
          count(buildingProfile:usage/text()) = 1"
91                          location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/2.0'][1]/*:cityObjectMember[
          namespace-uri()='http://www.opengis.net/citygml/2.0'][2]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
          building/2.0'][1]/*:boundedBy[namespace-uri()='http://www.opengis.net/citygml/building/2.0'][4]/*:WallSurface[
          namespace-uri()='http://www.opengis.net/citygml/building/2.0'][1]">
92          <svrl:text>There must be exactly one element of each attribute 'class', 'function', 'usage'.</svrl:text>
93    </svrl:failed-assert>
94    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:class"/>
95    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:function"/>
96    <svrl:fired-rule context="bldg:WallSurface"/>
97    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:class"/>
98    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:function"/>
99    <svrl:fired-rule context="bldg:WallSurface/buildingProfile:usage"/>
100 </svrl:schematron-output>
```

Listing A.4: validationReport.xml

B Gebäude-Profil CityGML 3.0

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- CityGML profile for the 'Building' module -->
<xs:schema xmlns="http://www.opengis.net/citygml/profiles/base/3.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/citygml/profiles/base/3.0" elementFormDefault="qualified" attributeFormDefault="
      unqualified" version="3.0.0">
  <xs:import namespace="http://www.opengis.net/citygml/building/3.0" schemaLocation="./building.xsd"/>
  <xs:import namespace="http://www.citygml.org/profile/building/clearanceSpaceComputation" schemaLocation="./BuildingADE.
      xsd"/>
</xs:schema>
```

Listing B.1: BuildingProfile.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://www.w3.org/2001/XMLSchema" xmlns:clrnceSpce="http://www.citygml
    .org/profile/building/clearanceSpaceComputation" xmlns:core="http://www.opengis.net/citygml/3.0" xmlns:gml="http://www
    .opengis.net/gml/3.2" elementFormDefault="qualified" targetNamespace="http://www.citygml.org/profile/building/
    clearanceSpaceComputation" version="">
  <annotation>
    <documentation>The Building package provides additional property classes to the CityGML-Building package in the context
       of a site plan.</documentation>
  </annotation>
  <import namespace="http://www.opengis.net/citygml/3.0" schemaLocation="./cityGMLBase.xsd"/>
  <import namespace="http://www.opengis.net/gml/3.2" schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
  <!--XML Schema document created by ShapeChange - http://shapechange.net/-->
  <element name="SurfaceProperties" substitutionGroup="core:ADEOfAbstractSpaceBoundary" type="
      clrnceSpce:SurfacePropertiesType">
    <annotation>
      <documentation>Additional properties of thematic surfaces providing information of the type, function or usage of the
       surface (e.g. facade surface, side wall, ...).</documentation>
    </annotation>
  </element>
  <complexType name="SurfacePropertiesType">
    <complexContent>
      <extension base="core:ADEOfAbstractSpaceBoundaryType">
        <sequence>
          <element minOccurs="0" name="class" type="gml:CodeType">
            <annotation>
              <documentation>Indicates the specific type of the thematic surface.</documentation>
            </annotation>
          </element>
          <element minOccurs="0" name="function" type="gml:CodeType">
            <annotation>
              <documentation>Specifies the intended purposes of the thematic surface.</documentation>
            </annotation>
          </element>
          <element minOccurs="0" name="usage" type="gml:CodeType">
            <annotation>
              <documentation>Specifies the actual uses of the thematic surface.</documentation>
            </annotation>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="SurfacePropertiesPropertyType">
    <sequence>
      <element ref="clrnceSpce:SurfaceProperties"/>
    </sequence>
  </complexType>
</schema>
```

Listing B.2: BuildingADE.xsd

```
<?xml version="1.0" encoding="UTF-8"?><schema xmlns="http://purl.oclc.org/dsdl/schematron" xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <title>Schematron constraints for schema 'Building'</title>
  <ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>
  <ns prefix="bldg" uri="http://www.opengis.net/citygml/building/3.0"/>
  <ns prefix="gml" uri="http://www.opengis.net/gml/3.2"/>
  <ns prefix="core" uri="http://www.opengis.net/citygml/3.0"/>
  <ns prefix="xlink" uri="http://www.w3.org/1999/xlink"/>
  <ns prefix="gml" uri="http://www.opengis.net/gml/3.2"/>
  <ns prefix="clrnceSpce" uri="http://www.citygml.org/profile/building/clearanceSpaceComputation"/>
  <pattern>
    <!-- Only allow one building element in the city model -->
  <rule context="core:CityModel">
    <assert test="count(./*) = 1 and count(core:cityObjectMember) = 1 and count(core:cityObjectMember/*) = 1 and count(
      core:cityObjectMember/bldg:Building) = 1">There is only allowed exactly element inside the city model which is of the
      type 'Building'.</assert>
  </rule>
  <!-- Only allow referenced objects, but not inline objects in CityObjectRelations -->
  <rule context="core:CityObjectRelation">
    <assert test="core:relatedTo/@xlink:href and count(core:relatedTo/text()) = 0 and count(core:relatedTo/*) = 0">Related
      element must be referenced by the use of xlink:href and must not be defined inline.</assert>
  </rule>
    <rule context="bldg:Building/bldg:class">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
    </rule>
    <rule context="bldg:BuildingPart/bldg:class">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
    </rule>
    <rule context="bldg:Building/bldg:function">
      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
```

```
35      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
36    </rule>
37    <rule context="bldg:BuildingPart/bldg:function">
38      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'</assert>
39      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
40      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
        GML 3.2 Dictionary</assert>
41      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
42    </rule>
43    <rule context="bldg:Building">
44      <assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">Each of the attributes 'class', '
        storeysAboveGround' must be present exactly once.</assert>
45      <assert test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(local-name()='
        RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()='
        GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
        core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
        namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])]">The bounding thematic
        surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one ADE element, which is of the type '
        SurfaceProperties'.</assert>
46      <assert test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]) &gt; 0">The
        element must be bounded by thematic surfaces.</assert>
47    </rule>
48    <rule context="bldg:BuildingInstallation">
49      <assert test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage/text()) = 1">Each
        of the attributes 'class', 'function', 'usage' must be present exactly once.</assert>
50      <assert test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(local-name()='
        RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()='
        GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
        core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
        namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])]">The bounding thematic
        surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one ADE element, which is of the type '
        SurfaceProperties'.</assert>
51      <assert test="core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]">The element must be
        bounded by thematic surfaces.</assert>
52    </rule>
53    <rule context="bldg:BuildingInstallation/bldg:class">
54      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'</assert>
55      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">Code list shall exist</assert>
56      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
        GML 3.2 Dictionary</assert>
57      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])">Code list value shall exist</assert>
58    </rule>
59    <rule context="bldg:BuildingInstallation/bldg:function">
60      <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'</assert>
61      <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
```

```
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')]))))">Code list shall exist</assert>
62  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
63  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
64  </rule>
65  <rule context="bldg:BuildingInstallation/bldg:usage">
66  <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'</assert>
67  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">Code list shall exist</assert>
68  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
69  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
70  </rule>
71  <rule context="bldg:BuildingPart">
72  <assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">Each of the attributes 'class', '
    storeysAboveGround' must be present exactly once.</assert>
73  <assert test="count(bldg:function/text()) = 1">The 'function' attribute must be present exactly once,</assert>
74  <assert test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(local-name()='
    RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()='
    GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
    core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
    namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])])">The bounding thematic
    surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one ADE element, which is of the type '
    SurfaceProperties'.</assert>
75  <assert test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]) &gt; 0">The
    element must be bounded by thematic surfaces.</assert>
76  </rule>
77  <rule context="clrnceSpce:SurfaceProperties">
78  <assert test="count(clrnceSpce:class/text()) = 1 and count(clrnceSpce:function/text()) = 1 and count(clrnceSpce:usage
    /text()) = 1">Each of the attributes 'class', 'function', 'usage' must be present exactly once.</assert>
79  </rule>
80  <rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class">
81  <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'</assert>
82  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">Code list shall exist</assert>
83  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
    GML 3.2 Dictionary</assert>
84  <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])">Code list value shall exist</assert>
85  </rule>
86  <rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function">
87  <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
```

```
      exampleCodeList_GML32.xml'</assert>
88    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
89    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
90    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
91    </rule>
92    <rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage">
93    <assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'">Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</assert>
94    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">Code list shall exist</assert>
95    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">Code list dictionary shall be represented using a
      GML 3.2 Dictionary</assert>
96    <assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])">Code list value shall exist</assert>
97    </rule>
98  </pattern>
99 </schema>
```

Listing B.3: schematron_building_profile_citygml3.0.sch

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsl:stylesheet xmlns:xs="http://www.w3.org/2001/XMLSchema"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:saxon="http://saxon.sf.net/"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:schold="http://www.ascc.net/xml/schematron"
                xmlns:iso="http://purl.oclc.org/dsdl/schematron"
                xmlns:xhtml="http://www.w3.org/1999/xhtml"
                xmlns:sch="http://purl.oclc.org/dsdl/schematron"
                xmlns:bldg="http://www.opengis.net/citygml/building/3.0"
                xmlns:gml="http://www.opengis.net/gml/3.2"
                xmlns:core="http://www.opengis.net/citygml/3.0"
                xmlns:xlink="http://www.w3.org/1999/xlink"
                xmlns:clrnceSpce="http://www.citygml.org/profile/building/clearanceSpaceComputation"
                version="2.0"><!--Implementers: please note that overriding process-prolog or process-root is
    the preferred method for meta-stylesheets to use where possible. -->
   <xsl:param name="archiveDirParameter"/>
   <xsl:param name="archiveNameParameter"/>
   <xsl:param name="fileNameParameter"/>
   <xsl:param name="fileDirParameter"/>
   <xsl:variable name="document-uri">
      <xsl:value-of select="document-uri(/)"/>
   </xsl:variable>
   <!--PHASES-->
   <!--PROLOG-->
   <xsl:output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
               method="xml"
               omit-xml-declaration="no"
               standalone="yes"
               indent="yes"/>
   <!--XSD TYPES FOR XSLT2-->
   <!--KEYS AND FUNCTIONS-->
   <!--DEFAULT RULES-->
   <!--MODE: SCHEMATRON-SELECT-FULL-PATH-->
   <!--This mode can be used to generate an ugly though full XPath for locators-->
   <xsl:template match="*" mode="schematron-select-full-path">
      <xsl:apply-templates select="." mode="schematron-get-full-path"/>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-->
   <!--This mode can be used to generate an ugly though full XPath for locators-->
   <xsl:template match="*" mode="schematron-get-full-path">
      <xsl:apply-templates select="parent::*" mode="schematron-get-full-path"/>
      <xsl:text>/</xsl:text>
      <xsl:choose>
         <xsl:when test="namespace-uri()=''">
            <xsl:value-of select="name()"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>*:</xsl:text>
            <xsl:value-of select="local-name()"/>
            <xsl:text>[namespace-uri()='</xsl:text>
            <xsl:value-of select="namespace-uri()"/>
            <xsl:text>']</xsl:text>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:variable name="preceding"
                    select="count(preceding-sibling::*[local-name()=local-name(current())
       and namespace-uri() = namespace-uri(current())])"/>
      <xsl:text>[</xsl:text>
      <xsl:value-of select="1+ $preceding"/>
      <xsl:text>]</xsl:text>
   </xsl:template>
   <xsl:template match="@*" mode="schematron-get-full-path">
      <xsl:apply-templates select="parent::*" mode="schematron-get-full-path"/>
      <xsl:text>/</xsl:text>
      <xsl:choose>
         <xsl:when test="namespace-uri()=''">@<xsl:value-of select="name()"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>@*[local-name()='</xsl:text>
            <xsl:value-of select="local-name()"/>
            <xsl:text>' and namespace-uri()='</xsl:text>
            <xsl:value-of select="namespace-uri()"/>
            <xsl:text>']</xsl:text>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-2-->
   <!--This mode can be used to generate prefixed XPath for humans-->
   <xsl:template match="node() | @*" mode="schematron-get-full-path-2">
      <xsl:for-each select="ancestor-or-self::*">
```

```
         <xsl:text>/</xsl:text>
         <xsl:value-of select="name(.)"/>
         <xsl:if test="preceding-sibling::*[name(.)=name(current())]">
            <xsl:text>[</xsl:text>
            <xsl:value-of select="count(preceding-sibling::*[name(.)=name(current())])+1"/>
            <xsl:text>]</xsl:text>
         </xsl:if>
      </xsl:for-each>
      <xsl:if test="not(self::*)">
         <xsl:text/>/@<xsl:value-of select="name(.)"/>
      </xsl:if>
   </xsl:template>
   <!--MODE: SCHEMATRON-FULL-PATH-3-->
   <!--This mode can be used to generate prefixed XPath for humans
  (Top-level element has index)-->
   <xsl:template match="node() | @*" mode="schematron-get-full-path-3">
      <xsl:for-each select="ancestor-or-self::*">
         <xsl:text>/</xsl:text>
         <xsl:value-of select="name(.)"/>
         <xsl:if test="parent::*">
            <xsl:text>[</xsl:text>
            <xsl:value-of select="count(preceding-sibling::*[name(.)=name(current())])+1"/>
            <xsl:text>]</xsl:text>
         </xsl:if>
      </xsl:for-each>
      <xsl:if test="not(self::*)">
         <xsl:text/>/@<xsl:value-of select="name(.)"/>
      </xsl:if>
   </xsl:template>
   <!--MODE: GENERATE-ID-FROM-PATH -->
   <xsl:template match="/" mode="generate-id-from-path"/>
   <xsl:template match="text()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.text-', 1+count(preceding-sibling::text()), '-')"/>
   </xsl:template>
   <xsl:template match="comment()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.comment-', 1+count(preceding-sibling::comment()), '-')"/>
   </xsl:template>
   <xsl:template match="processing-instruction()" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.processing-instruction-', 1+count(preceding-sibling::processing-instruction()), '-')"/
      >
   </xsl:template>
   <xsl:template match="@*" mode="generate-id-from-path">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:value-of select="concat('.@', name())"/>
   </xsl:template>
   <xsl:template match="*" mode="generate-id-from-path" priority="-0.5">
      <xsl:apply-templates select="parent::*" mode="generate-id-from-path"/>
      <xsl:text>.</xsl:text>
      <xsl:value-of select="concat('.',name(),'-',1+count(preceding-sibling::*[name()=name(current())]),'-')"/>
   </xsl:template>
   <!--MODE: GENERATE-ID-2 -->
   <xsl:template match="/" mode="generate-id-2">U</xsl:template>
   <xsl:template match="*" mode="generate-id-2" priority="2">
      <xsl:text>U</xsl:text>
      <xsl:number level="multiple" count="*"/>
   </xsl:template>
   <xsl:template match="node()" mode="generate-id-2">
      <xsl:text>U.</xsl:text>
      <xsl:number level="multiple" count="*"/>
      <xsl:text>n</xsl:text>
      <xsl:number count="node()"/>
   </xsl:template>
   <xsl:template match="@*" mode="generate-id-2">
      <xsl:text>U.</xsl:text>
      <xsl:number level="multiple" count="*"/>
      <xsl:text>_</xsl:text>
      <xsl:value-of select="string-length(local-name(.))"/>
      <xsl:text>_</xsl:text>
      <xsl:value-of select="translate(name(),':','.')"/>
   </xsl:template>
   <!--Strip characters-->
   <xsl:template match="text()" priority="-1"/>
   <!--SCHEMA SETUP-->
   <xsl:template match="/">
      <svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                              title="Schematron constraints for schema 'Building'"
                              schemaVersion="">
         <xsl:comment>
            <xsl:value-of select="$archiveDirParameter"/>
```

```
    <xsl:value-of select="$archiveNameParameter"/>
    <xsl:value-of select="$fileNameParameter"/>
    <xsl:value-of select="$fileDirParameter"/>
        </xsl:comment>
        <svrl:ns-prefix-in-attribute-values uri="http://purl.oclc.org/dsdl/schematron" prefix="sch"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/building/3.0" prefix="bldg"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml/3.2" prefix="gml"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/3.0" prefix="core"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.w3.org/1999/xlink" prefix="xlink"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml/3.2" prefix="gml"/>
        <svrl:ns-prefix-in-attribute-values uri="http://www.citygml.org/profile/building/clearanceSpaceComputation"
                                            prefix="clrnceSpce"/>
        <svrl:active-pattern>
           <xsl:attribute name="document">
              <xsl:value-of select="document-uri(/)"/>
           </xsl:attribute>
           <xsl:apply-templates/>
        </svrl:active-pattern>
        <xsl:apply-templates select="/" mode="M8"/>
     </svrl:schematron-output>
  </xsl:template>
  <!--SCHEMATRON PATTERNS-->
  <svrl:text xmlns:svrl="http://purl.oclc.org/dsdl/svrl">Schematron constraints for schema 'Building'</svrl:text>
  <!--PATTERN -->
  <!--RULE -->
  <xsl:template match="core:CityModel" priority="1015" mode="M8">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="core:CityModel"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="count(./*) = 1 and count(core:cityObjectMember) = 1 and count(core:cityObjectMember/*) = 1 and
     count(core:cityObjectMember/bldg:Building) = 1"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="count(./*) = 1 and count(core:cityObjectMember) = 1 and count(core:cityObjectMember
     /*) = 1 and count(core:cityObjectMember/bldg:Building) = 1">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>There is only allowed exactly element inside the city model which is of the type 'Building'.</
     svrl:text>
           </svrl:failed-assert>
        </xsl:otherwise>
     </xsl:choose>
     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
  </xsl:template>
  <!--RULE -->
  <xsl:template match="core:CityObjectRelation" priority="1014" mode="M8">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                      context="core:CityObjectRelation"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="core:relatedTo/@xlink:href and count(core:relatedTo/text()) = 0 and count(core:relatedTo/*) = 0"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="core:relatedTo/@xlink:href and count(core:relatedTo/text()) = 0 and count(
     core:relatedTo/*) = 0">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>Related element must be referenced by the use of xlink:href and must not be defined inline.</
     svrl:text>
           </svrl:failed-assert>
        </xsl:otherwise>
     </xsl:choose>
     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
  </xsl:template>
  <!--RULE -->
  <xsl:template match="bldg:Building/bldg:class" priority="1013" mode="M8">
     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                      context="bldg:Building/bldg:class"/>
     <!--ASSERT -->
     <xsl:choose>
        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
        <xsl:otherwise>
           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
              <xsl:attribute name="location">
                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
              </xsl:attribute>
              <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
```

```
      exampleCodeList_GML32.xml'</svrl:text>
237           </svrl:failed-assert>
238        </xsl:otherwise>
239     </xsl:choose>
240     <!--ASSERT -->
241     <xsl:choose>
242        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
       codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
       codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
       codeListUrl,'#')]))"/>
243        <xsl:otherwise>
244           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
245                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
       ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
       $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
       codeListUrl,'#')]))">
246              <xsl:attribute name="location">
247                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
248              </xsl:attribute>
249              <svrl:text>Code list shall exist</svrl:text>
250           </svrl:failed-assert>
251        </xsl:otherwise>
252     </xsl:choose>
253     <!--ASSERT -->
254     <xsl:choose>
255        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
       codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
       ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
       //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))"/>
256        <xsl:otherwise>
257           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
258                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
       ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
       substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
       exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
259              <xsl:attribute name="location">
260                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
261              </xsl:attribute>
262              <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
263           </svrl:failed-assert>
264        </xsl:otherwise>
265     </xsl:choose>
266     <!--ASSERT -->
267     <xsl:choose>
268        <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
       and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
       codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
       substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
       [gml:identifier = $codeListValue])"/>
269        <xsl:otherwise>
270           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
271                               test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
       codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
       gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
       ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
       gml:Definition[gml:identifier = $codeListValue])">
272              <xsl:attribute name="location">
273                 <xsl:apply-templates select="." mode="schematron-select-full-path"/>
274              </xsl:attribute>
275              <svrl:text>Code list value shall exist</svrl:text>
276           </svrl:failed-assert>
277        </xsl:otherwise>
278     </xsl:choose>
279     <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
280  </xsl:template>
281  <!--RULE -->
282  <xsl:template match="bldg:BuildingPart/bldg:class" priority="1012" mode="M8">
283     <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
284                      context="bldg:BuildingPart/bldg:class"/>
285     <!--ASSERT -->
286     <xsl:choose>
287        <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
288        <xsl:otherwise>
289           <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
290                               test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
```

```
        SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
291             <xsl:attribute name="location">
292                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
293             </xsl:attribute>
294             <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
        exampleCodeList_GML32.xml'</svrl:text>
295          </svrl:failed-assert>
296       </xsl:otherwise>
297    </xsl:choose>
298    <!--ASSERT -->
299    <xsl:choose>
300       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
        codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')]))) "/>
301       <xsl:otherwise>
302          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
303                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
        Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
        ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
        $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
        codeListUrl,'#')])))">
304             <xsl:attribute name="location">
305                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
306             </xsl:attribute>
307             <svrl:text>Code list shall exist</svrl:text>
308          </svrl:failed-assert>
309       </xsl:otherwise>
310    </xsl:choose>
311    <!--ASSERT -->
312    <xsl:choose>
313       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
        codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) "/>
314       <xsl:otherwise>
315          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
316                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
        Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
        ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
        substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
        exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
317             <xsl:attribute name="location">
318                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
319             </xsl:attribute>
320             <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
321          </svrl:failed-assert>
322       </xsl:otherwise>
323    </xsl:choose>
324    <!--ASSERT -->
325    <xsl:choose>
326       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
        codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])"/>
327       <xsl:otherwise>
328          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
329                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
        Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
        codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
        gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
        ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
        gml:Definition[gml:identifier = $codeListValue])">
330             <xsl:attribute name="location">
331                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
332             </xsl:attribute>
333             <svrl:text>Code list value shall exist</svrl:text>
334          </svrl:failed-assert>
335       </xsl:otherwise>
336    </xsl:choose>
337    <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
338 </xsl:template>
339 <!--RULE -->
340 <xsl:template match="bldg:Building/bldg:function" priority="1011" mode="M8">
341    <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
342                     context="bldg:Building/bldg:function"/>
343    <!--ASSERT -->
344    <xsl:choose>
```

```
345          <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
346          <xsl:otherwise>
347             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
348                                 test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
349                <xsl:attribute name="location">
350                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
351                </xsl:attribute>
352                <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
     exampleCodeList_GML32.xml'</svrl:text>
353             </svrl:failed-assert>
354          </xsl:otherwise>
355       </xsl:choose>
356       <!--ASSERT -->
357       <xsl:choose>
358          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
     codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
     codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
     codeListUrl,'#')])))"/>
359          <xsl:otherwise>
360             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
361                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
     Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
     ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
     $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
     codeListUrl,'#')])))">
362                <xsl:attribute name="location">
363                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
364                </xsl:attribute>
365                <svrl:text>Code list shall exist</svrl:text>
366             </svrl:failed-assert>
367          </xsl:otherwise>
368       </xsl:choose>
369       <!--ASSERT -->
370       <xsl:choose>
371          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
     codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
     ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
     //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
372          <xsl:otherwise>
373             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
374                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
     Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
     ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
     substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
     exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
375                <xsl:attribute name="location">
376                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
377                </xsl:attribute>
378                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
379             </svrl:failed-assert>
380          </xsl:otherwise>
381       </xsl:choose>
382       <!--ASSERT -->
383       <xsl:choose>
384          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
     SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
     and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
     codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
     substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
     [gml:identifier = $codeListValue])"/>
385          <xsl:otherwise>
386             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
387                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
     Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
     codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
     gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
     ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
     gml:Definition[gml:identifier = $codeListValue])">
388                <xsl:attribute name="location">
389                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
390                </xsl:attribute>
391                <svrl:text>Code list value shall exist</svrl:text>
392             </svrl:failed-assert>
393          </xsl:otherwise>
394       </xsl:choose>
395       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
396    </xsl:template>
397    <!--RULE -->
```

```
<xsl:template match="bldg:BuildingPart/bldg:function" priority="1010" mode="M8">
   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                    context="bldg:BuildingPart/bldg:function"/>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
   exampleCodeList_GML32.xml'</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
   codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
   $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
   codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
   codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
   ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
   //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
   ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
   substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
   exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
   SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
   and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
   codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
   substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
   [gml:identifier = $codeListValue])"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
   Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
   codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
   gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
   ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
   gml:Definition[gml:identifier = $codeListValue])">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list value shall exist</svrl:text>
         </svrl:failed-assert>
```

```
451         </xsl:otherwise>
452      </xsl:choose>
453      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
454   </xsl:template>
455   <!--RULE -->
456   <xsl:template match="bldg:Building" priority="1009" mode="M8">
457      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:Building"/>
458      <!--ASSERT -->
459      <xsl:choose>
460         <xsl:when test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"/>
461         <xsl:otherwise>
462            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
463                                test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">
464               <xsl:attribute name="location">
465                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
466               </xsl:attribute>
467               <svrl:text>Each of the attributes 'class' , 'storeysAboveGround' must be present exactly once.</svrl:text>
468            </svrl:failed-assert>
469         </xsl:otherwise>
470      </xsl:choose>
471      <!--ASSERT -->
472      <xsl:choose>
473         <xsl:when test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(
      local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()
      ='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
      core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
      namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])])"/>
474         <xsl:otherwise>
475            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
476                                test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[
      not(local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(
      local-name()='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
      core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
      namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])])">
477               <xsl:attribute name="location">
478                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
479               </xsl:attribute>
480               <svrl:text>The bounding thematic surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one
      ADE element, which is of the type 'SurfaceProperties'.</svrl:text>
481            </svrl:failed-assert>
482         </xsl:otherwise>
483      </xsl:choose>
484      <!--ASSERT -->
485      <xsl:choose>
486         <xsl:when test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]) &gt; 0"/>
487         <xsl:otherwise>
488            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
489                                test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])
       &gt; 0">
490               <xsl:attribute name="location">
491                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
492               </xsl:attribute>
493               <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
494            </svrl:failed-assert>
495         </xsl:otherwise>
496      </xsl:choose>
497      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
498   </xsl:template>
499   <!--RULE -->
500   <xsl:template match="bldg:BuildingInstallation" priority="1008" mode="M8">
501      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
502                       context="bldg:BuildingInstallation"/>
503      <!--ASSERT -->
504      <xsl:choose>
505         <xsl:when test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage/text()) = 1"
      />
506         <xsl:otherwise>
507            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
508                                test="count(bldg:class/text()) = 1 and count(bldg:function/text()) = 1 and count(bldg:usage
      /text()) = 1">
509               <xsl:attribute name="location">
510                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
511               </xsl:attribute>
512               <svrl:text>Each of the attributes 'class', 'function', 'usage' must be present exactly once.</svrl:text>
513            </svrl:failed-assert>
514         </xsl:otherwise>
515      </xsl:choose>
516      <!--ASSERT -->
517      <xsl:choose>
518         <xsl:when test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(
      local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()
      ='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
```

```
      core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
      namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])"/>
519       <xsl:otherwise>
520          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
521                              test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[
      not(local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(
      local-name()='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
      core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
      namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])">
522             <xsl:attribute name="location">
523                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
524             </xsl:attribute>
525             <svrl:text>The bounding thematic surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one
      ADE element, which is of the type 'SurfaceProperties'.</svrl:text>
526          </svrl:failed-assert>
527       </xsl:otherwise>
528    </xsl:choose>
529    <!--ASSERT -->
530    <xsl:choose>
531       <xsl:when test="core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]"/>
532       <xsl:otherwise>
533          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
534                              test="core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]">
535             <xsl:attribute name="location">
536                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
537             </xsl:attribute>
538             <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
539          </svrl:failed-assert>
540       </xsl:otherwise>
541    </xsl:choose>
542    <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
543 </xsl:template>
544 <!--RULE -->
545 <xsl:template match="bldg:BuildingInstallation/bldg:class"
546               priority="1007"
547               mode="M8">
548    <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
549                     context="bldg:BuildingInstallation/bldg:class"/>
550    <!--ASSERT -->
551    <xsl:choose>
552       <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
553       <xsl:otherwise>
554          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
555                              test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
556             <xsl:attribute name="location">
557                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
558             </xsl:attribute>
559             <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</svrl:text>
560          </svrl:failed-assert>
561       </xsl:otherwise>
562    </xsl:choose>
563    <!--ASSERT -->
564    <xsl:choose>
565       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
566       <xsl:otherwise>
567          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
568                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
569             <xsl:attribute name="location">
570                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
571             </xsl:attribute>
572             <svrl:text>Code list shall exist</svrl:text>
573          </svrl:failed-assert>
574       </xsl:otherwise>
575    </xsl:choose>
576    <!--ASSERT -->
577    <xsl:choose>
578       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
```

```
579         <xsl:otherwise>
580            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
581                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
582               <xsl:attribute name="location">
583                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
584               </xsl:attribute>
585               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
586            </svrl:failed-assert>
587         </xsl:otherwise>
588      </xsl:choose>
589      <!--ASSERT -->
590      <xsl:choose>
591         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])"/>
592         <xsl:otherwise>
593            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
594                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
    codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
    gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
    ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
    gml:Definition[gml:identifier = $codeListValue])">
595               <xsl:attribute name="location">
596                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
597               </xsl:attribute>
598               <svrl:text>Code list value shall exist</svrl:text>
599            </svrl:failed-assert>
600         </xsl:otherwise>
601      </xsl:choose>
602      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
603   </xsl:template>
604   <!--RULE -->
605   <xsl:template match="bldg:BuildingInstallation/bldg:function"
606                 priority="1006"
607                 mode="M8">
608      <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
609                       context="bldg:BuildingInstallation/bldg:function"/>
610      <!--ASSERT -->
611      <xsl:choose>
612         <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
613         <xsl:otherwise>
614            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
615                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
616               <xsl:attribute name="location">
617                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
618               </xsl:attribute>
619               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'</svrl:text>
620            </svrl:failed-assert>
621         </xsl:otherwise>
622      </xsl:choose>
623      <!--ASSERT -->
624      <xsl:choose>
625         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))"/>
626         <xsl:otherwise>
627            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
628                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
    $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')])))">
629               <xsl:attribute name="location">
630                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
631               </xsl:attribute>
632               <svrl:text>Code list shall exist</svrl:text>
633            </svrl:failed-assert>
634         </xsl:otherwise>
635      </xsl:choose>
```

```
636         <!--ASSERT -->
637         <xsl:choose>
638            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
            ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
            //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
639            <xsl:otherwise>
640               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
641                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
            substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
            exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
642                  <xsl:attribute name="location">
643                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
644                  </xsl:attribute>
645                  <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
646               </svrl:failed-assert>
647            </xsl:otherwise>
648         </xsl:choose>
649         <!--ASSERT -->
650         <xsl:choose>
651            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
            and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
            codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
            substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
            [gml:identifier = $codeListValue])"/>
652            <xsl:otherwise>
653               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
654                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
            codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
            gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
            ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
            gml:Definition[gml:identifier = $codeListValue])">
655                  <xsl:attribute name="location">
656                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
657                  </xsl:attribute>
658                  <svrl:text>Code list value shall exist</svrl:text>
659               </svrl:failed-assert>
660            </xsl:otherwise>
661         </xsl:choose>
662         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
663      </xsl:template>
664      <!--RULE -->
665      <xsl:template match="bldg:BuildingInstallation/bldg:usage"
666                    priority="1005"
667                    mode="M8">
668         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
669                          context="bldg:BuildingInstallation/bldg:usage"/>
670         <!--ASSERT -->
671         <xsl:choose>
672            <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
673            <xsl:otherwise>
674               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
675                                   test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
676                  <xsl:attribute name="location">
677                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
678                  </xsl:attribute>
679                  <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
            exampleCodeList_GML32.xml'</svrl:text>
680               </svrl:failed-assert>
681            </xsl:otherwise>
682         </xsl:choose>
683         <!--ASSERT -->
684         <xsl:choose>
685            <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
            codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
            codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')])))"/>
686            <xsl:otherwise>
687               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
688                                   test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
            Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
            ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
            $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
            codeListUrl,'#')])))">
```

```
689             <xsl:attribute name="location">
690                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
691             </xsl:attribute>
692             <svrl:text>Code list shall exist</svrl:text>
693          </svrl:failed-assert>
694       </xsl:otherwise>
695    </xsl:choose>
696    <!--ASSERT -->
697    <xsl:choose>
698       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
       codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
       ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
       //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
699       <xsl:otherwise>
700          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
701                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
       ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
       substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
       exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))">
702             <xsl:attribute name="location">
703                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
704             </xsl:attribute>
705             <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
706          </svrl:failed-assert>
707       </xsl:otherwise>
708    </xsl:choose>
709    <!--ASSERT -->
710    <xsl:choose>
711       <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
       and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
       codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
       substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
       [gml:identifier = $codeListValue])"/>
712       <xsl:otherwise>
713          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
714                              test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
       Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
       codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
       gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
       ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
       gml:Definition[gml:identifier = $codeListValue])">
715             <xsl:attribute name="location">
716                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
717             </xsl:attribute>
718             <svrl:text>Code list value shall exist</svrl:text>
719          </svrl:failed-assert>
720       </xsl:otherwise>
721    </xsl:choose>
722    <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
723 </xsl:template>
724 <!--RULE -->
725 <xsl:template match="bldg:BuildingPart" priority="1004" mode="M8">
726    <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl" context="bldg:BuildingPart"/>
727    <!--ASSERT -->
728    <xsl:choose>
729       <xsl:when test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"/>
730       <xsl:otherwise>
731          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
732                              test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0">
733             <xsl:attribute name="location">
734                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
735             </xsl:attribute>
736             <svrl:text>Each of the attributes 'class', 'storeysAboveGround' must be present exactly once.</svrl:text>
737          </svrl:failed-assert>
738       </xsl:otherwise>
739    </xsl:choose>
740    <!--ASSERT -->
741    <xsl:choose>
742       <xsl:when test="count(bldg:function/text()) = 1"/>
743       <xsl:otherwise>
744          <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
745                              test="count(bldg:function/text()) = 1">
746             <xsl:attribute name="location">
747                <xsl:apply-templates select="." mode="schematron-select-full-path"/>
748             </xsl:attribute>
749             <svrl:text>The 'function' attribute must be present exactly once,</svrl:text>
750          </svrl:failed-assert>
751       </xsl:otherwise>
752    </xsl:choose>
```

```
753         <!--ASSERT -->
754         <xsl:choose>
755            <xsl:when test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(
            local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()
            ='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
            core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
            namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])])"/>
756            <xsl:otherwise>
757               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
758                                   test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[
            not(local-name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(
            local-name()='GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(
            core:adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
            namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])]">
759                  <xsl:attribute name="location">
760                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
761                  </xsl:attribute>
762                  <svrl:text>The bounding thematic surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one
            ADE element, which is of the type 'SurfaceProperties'.</svrl:text>
763               </svrl:failed-assert>
764            </xsl:otherwise>
765         </xsl:choose>
766         <!--ASSERT -->
767         <xsl:choose>
768            <xsl:when test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]) &gt; 0"/>
769            <xsl:otherwise>
770               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
771                                   test="count(core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])
             &gt; 0">
772                  <xsl:attribute name="location">
773                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
774                  </xsl:attribute>
775                  <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
776               </svrl:failed-assert>
777            </xsl:otherwise>
778         </xsl:choose>
779         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
780      </xsl:template>
781      <!--RULE -->
782      <xsl:template match="clrnceSpce:SurfaceProperties" priority="1003" mode="M8">
783         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
784                          context="clrnceSpce:SurfaceProperties"/>
785         <!--ASSERT -->
786         <xsl:choose>
787            <xsl:when test="count(clrnceSpce:class/text()) = 1 and count(clrnceSpce:function/text()) = 1 and count(
            clrnceSpce:usage/text()) = 1"/>
788            <xsl:otherwise>
789               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
790                                   test="count(clrnceSpce:class/text()) = 1 and count(clrnceSpce:function/text()) = 1 and
            count(clrnceSpce:usage/text()) = 1">
791                  <xsl:attribute name="location">
792                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
793                  </xsl:attribute>
794                  <svrl:text>Each of the attributes 'class', 'function', 'usage' must be present exactly once.</svrl:text>
795               </svrl:failed-assert>
796            </xsl:otherwise>
797         </xsl:choose>
798         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
799      </xsl:template>
800      <!--RULE -->
801      <xsl:template match="clrnceSpce:SurfaceProperties/clrnceSpce:class"
802                    priority="1002"
803                    mode="M8">
804         <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
805                          context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
806         <!--ASSERT -->
807         <xsl:choose>
808            <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
809            <xsl:otherwise>
810               <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
811                                   test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
            SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
812                  <xsl:attribute name="location">
813                     <xsl:apply-templates select="." mode="schematron-select-full-path"/>
814                  </xsl:attribute>
815                  <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
            exampleCodeList_GML32.xml'</svrl:text>
816               </svrl:failed-assert>
817            </xsl:otherwise>
818         </xsl:choose>
819         <!--ASSERT -->
```

```
820       <xsl:choose>
821          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
          codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')]))) "/>
822          <xsl:otherwise>
823             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
824                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
          $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
          codeListUrl,'#')]))) ">
825                <xsl:attribute name="location">
826                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
827                </xsl:attribute>
828                <svrl:text>Code list shall exist</svrl:text>
829             </svrl:failed-assert>
830          </xsl:otherwise>
831       </xsl:choose>
832       <!--ASSERT -->
833       <xsl:choose>
834          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
          codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
          ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
          //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
835          <xsl:otherwise>
836             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
837                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
          ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
          substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
          exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
838                <xsl:attribute name="location">
839                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
840                </xsl:attribute>
841                <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
842             </svrl:failed-assert>
843          </xsl:otherwise>
844       </xsl:choose>
845       <!--ASSERT -->
846       <xsl:choose>
847          <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
          and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
          codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
          substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
          [gml:identifier = $codeListValue])"/>
848          <xsl:otherwise>
849             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
850                                 test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
          Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
          codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
          gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
          ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
          gml:Definition[gml:identifier = $codeListValue])">
851                <xsl:attribute name="location">
852                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
853                </xsl:attribute>
854                <svrl:text>Code list value shall exist</svrl:text>
855             </svrl:failed-assert>
856          </xsl:otherwise>
857       </xsl:choose>
858       <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
859    </xsl:template>
860    <!--RULE -->
861    <xsl:template match="clrnceSpce:SurfaceProperties/clrnceSpce:function"
862                  priority="1001"
863                  mode="M8">
864       <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
865                        context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
866       <!--ASSERT -->
867       <xsl:choose>
868          <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
869          <xsl:otherwise>
870             <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
871                                 test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
          SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
872                <xsl:attribute name="location">
873                   <xsl:apply-templates select="." mode="schematron-select-full-path"/>
```

```
            </xsl:attribute>
            <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
      exampleCodeList_GML32.xml'</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
      codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
      $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
      codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
      codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
      ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
      //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
      ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
      substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
      exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
      and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
      codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"/>
      <xsl:otherwise>
         <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                             test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
      Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
      codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
      gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
      ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
      gml:Definition[gml:identifier = $codeListValue])">
            <xsl:attribute name="location">
               <xsl:apply-templates select="." mode="schematron-select-full-path"/>
            </xsl:attribute>
            <svrl:text>Code list value shall exist</svrl:text>
         </svrl:failed-assert>
      </xsl:otherwise>
   </xsl:choose>
   <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
</xsl:template>
<!--RULE -->
<xsl:template match="clrnceSpce:SurfaceProperties/clrnceSpce:usage"
              priority="1000"
              mode="M8">
   <svrl:fired-rule xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                    context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <!--ASSERT -->
   <xsl:choose>
      <xsl:when test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
```

```
SaxonXLSTProcessor/exampleCodeList_GML32.xml'"/>
929         <xsl:otherwise>
930            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
931                                test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml'">
932               <xsl:attribute name="location">
933                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
934               </xsl:attribute>
935               <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/
    exampleCodeList_GML32.xml'</svrl:text>
936            </svrl:failed-assert>
937         </xsl:otherwise>
938      </xsl:choose>
939      <!--ASSERT -->
940      <xsl:choose>
941         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for $
    codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')]))) "/>
942         <xsl:otherwise>
943            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
944                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl)) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and boolean(for
    $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc//*[@gml:id = substring-after($
    codeListUrl,'#')]))) ">
945               <xsl:attribute name="location">
946                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
947               </xsl:attribute>
948               <svrl:text>Code list shall exist</svrl:text>
949            </svrl:failed-assert>
950         </xsl:otherwise>
951      </xsl:choose>
952      <!--ASSERT -->
953      <xsl:choose>
954         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available($
    codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) "/>
955         <xsl:otherwise>
956            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
957                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available
    ($codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(
    substring-before($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return
    exists($codelistDoc//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))) ">
958               <xsl:attribute name="location">
959                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
960               </xsl:attribute>
961               <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
962            </svrl:failed-assert>
963         </xsl:otherwise>
964      </xsl:choose>
965      <!--ASSERT -->
966      <xsl:choose>
967         <xsl:when test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier = $
    codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])"/>
968         <xsl:otherwise>
969            <svrl:failed-assert xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
970                                test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/
    Documents/SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($
    codeListUrl, '#')) and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[
    gml:identifier = $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')
    ) and doc(substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/
    gml:Definition[gml:identifier = $codeListValue])">
971               <xsl:attribute name="location">
972                  <xsl:apply-templates select="." mode="schematron-select-full-path"/>
973               </xsl:attribute>
974               <svrl:text>Code list value shall exist</svrl:text>
975            </svrl:failed-assert>
976         </xsl:otherwise>
977      </xsl:choose>
978      <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
979   </xsl:template>
980   <xsl:template match="text()" priority="-1" mode="M8"/>
981   <xsl:template match="@*|node()" priority="-2" mode="M8">
```

```
982         <xsl:apply-templates select="*|comment()|processing-instruction()" mode="M8"/>
983     </xsl:template>
984 </xsl:stylesheet>
```

Listing B.4: validationScriptCityGML3.0.xsl

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                        xmlns:saxon="http://saxon.sf.net/"
                        xmlns:schold="http://www.ascc.net/xml/schematron"
                        xmlns:iso="http://purl.oclc.org/dsdl/schematron"
                        xmlns:xhtml="http://www.w3.org/1999/xhtml"
                        xmlns:sch="http://purl.oclc.org/dsdl/schematron"
                        xmlns:bldg="http://www.opengis.net/citygml/building/3.0"
                        xmlns:gml="http://www.opengis.net/gml/3.2"
                        xmlns:core="http://www.opengis.net/citygml/3.0"
                        xmlns:xlink="http://www.w3.org/1999/xlink"
                        xmlns:clrnceSpce="http://www.citygml.org/profile/building/clearanceSpaceComputation"
                        title="Schematron constraints for schema 'Building'"
            schemaVersion="">
   <svrl:ns-prefix-in-attribute-values uri="http://purl.oclc.org/dsdl/schematron" prefix="sch"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/building/3.0" prefix="bldg"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml/3.2" prefix="gml"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/citygml/3.0" prefix="core"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.w3.org/1999/xlink" prefix="xlink"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.opengis.net/gml/3.2" prefix="gml"/>
   <svrl:ns-prefix-in-attribute-values uri="http://www.citygml.org/profile/building/clearanceSpaceComputation"
                                       prefix="clrnceSpce"/>
   <svrl:active-pattern document="file:/C:/Users/felix/Documents/SaxonXLSTProcessor/./data/building.gml"/>
   <svrl:fired-rule context="core:CityModel"/>
   <svrl:failed-assert test="count(./*) = 1 and count(core:cityObjectMember) = 1 and count(core:cityObjectMember/*) = 1 and
      count(core:cityObjectMember/bldg:Building) = 1"
                       location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]">
      <svrl:text>There is only allowed exactly element inside the city model which is of the type 'Building'.</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="bldg:Building"/>
   <svrl:failed-assert test="count(bldg:class/text()) = 1 and bldg:storeysAboveGround &gt;= 0"
                       location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/3.0'][1]">
      <svrl:text>Each of the attributes 'class' , 'storeysAboveGround' must be present exactly once.</svrl:text>
   </svrl:failed-assert>
   <svrl:failed-assert test="not((core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href])[not(local
      -name()='RoofSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and not(local-name()='
      GroundSurface' and namespace-uri()='http://www.opengis.net/citygml/construction/3.0') and (count(core:
      adeOfAbstractSpaceBoundary/*) != 1 or core:adeOfAbstractSpaceBoundary/*[not(local-name()='SurfaceProperties' and
      namespace-uri()='http://www.citygml.org/profile/building/clearanceSpaceComputation')])])"
                       location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/3.0'][1]">
      <svrl:text>The bounding thematic surfaces (except 'RoofSurface' and 'GroundSurface') must have exactly one ADE
      element, which is of the type 'SurfaceProperties'.</svrl:text>
   </svrl:failed-assert>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
   <svrl:fired-rule context="bldg:Building/bldg:class"/>
   <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
      SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
                       location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/3.0'][1]/*:class[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]">
      <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
      xml'</svrl:text>
```

```
68    </svrl:failed-assert>
69    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
         $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
         ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
         //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
70                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:class[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]">
71       <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
72    </svrl:failed-assert>
73    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
         and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
         $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
         substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
         [gml:identifier = $codeListValue])"
74                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:class[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]">
75       <svrl:text>Code list value shall exist</svrl:text>
76    </svrl:failed-assert>
77    <svrl:fired-rule context="bldg:BuildingInstallation"/>
78    <svrl:failed-assert test="core:boundary/* | //*[concat('#',@gml:id)=current()/core:boundary/@xlink:href]"
79                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
         BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]">
80       <svrl:text>The element must be bounded by thematic surfaces.</svrl:text>
81    </svrl:failed-assert>
82    <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
83    <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
84                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
         BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
         ://www.opengis.net/citygml/building/3.0'][1]">
85       <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
         xml'</svrl:text>
86    </svrl:failed-assert>
87    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
         $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
         ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
         //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))"
88                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
         BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
         ://www.opengis.net/citygml/building/3.0'][1]">
89       <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
90    </svrl:failed-assert>
91    <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
         and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
         $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
         substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
         [gml:identifier = $codeListValue])"
92                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
         BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
         ://www.opengis.net/citygml/building/3.0'][1]">
93       <svrl:text>Code list value shall exist</svrl:text>
94    </svrl:failed-assert>
95    <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
96    <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
97                    location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
         namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
         building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
         BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:function[namespace-uri()='
         http://www.opengis.net/citygml/building/3.0'][1]">
98       <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
         xml'</svrl:text>
99    </svrl:failed-assert>
100   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
         SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
         $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
         ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
         //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')]))"
```

```
101                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:function[namespace-uri()='
    http://www.opengis.net/citygml/building/3.0'][1]">
102         <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
103     </svrl:failed-assert>
104     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
    $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])"
105                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:function[namespace-uri()='
    http://www.opengis.net/citygml/building/3.0'][1]">
106         <svrl:text>Code list value shall exist</svrl:text>
107     </svrl:failed-assert>
108     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
109     <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
110                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:usage[namespace-uri()='http
    ://www.opengis.net/citygml/building/3.0'][1]">
111         <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
    xml'</svrl:text>
112     </svrl:failed-assert>
113     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
    $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
    ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
    //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
114                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:usage[namespace-uri()='http
    ://www.opengis.net/citygml/building/3.0'][1]">
115         <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
116     </svrl:failed-assert>
117     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
    and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
    $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
    substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
    [gml:identifier = $codeListValue])"
118                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:usage[namespace-uri()='http
    ://www.opengis.net/citygml/building/3.0'][1]">
119         <svrl:text>Code list value shall exist</svrl:text>
120     </svrl:failed-assert>
121     <svrl:fired-rule context="bldg:BuildingInstallation"/>
122     <svrl:fired-rule context="core:CityObjectRelation"/>
123     <svrl:failed-assert test="core:relatedTo/@xlink:href and count(core:relatedTo/text()) = 0 and count(core:relatedTo/*) =
    0"
124                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][2]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:relatedTo[namespace-uri()='
    http://www.opengis.net/citygml/3.0'][1]/*:CityObjectRelation[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]"
    >
125         <svrl:text>Related element must be referenced by the use of xlink:href and must not be defined inline.</svrl:text>
126     </svrl:failed-assert>
127     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
128     <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
129                         location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
    namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
    building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][2]/*:
    BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
    ://www.opengis.net/citygml/building/3.0'][1]">
130         <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
    xml'</svrl:text>
131     </svrl:failed-assert>
132     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
    SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
    $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
```

```
($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
133                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][2]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
134     <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
135   </svrl:failed-assert>
136   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
$codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
[gml:identifier = $codeListValue])"
137                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][2]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
138     <svrl:text>Code list value shall exist</svrl:text>
139   </svrl:failed-assert>
140   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
141   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
142   <svrl:fired-rule context="bldg:BuildingInstallation"/>
143   <svrl:fired-rule context="core:CityObjectRelation"/>
144   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
145   <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
146                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][3]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
147     <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
xml'</svrl:text>
148   </svrl:failed-assert>
149   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
$codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
150                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][3]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
151     <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
152   </svrl:failed-assert>
153   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
$codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
[gml:identifier = $codeListValue])"
154                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][3]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
155     <svrl:text>Code list value shall exist</svrl:text>
156   </svrl:failed-assert>
157   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
158   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
159   <svrl:fired-rule context="bldg:BuildingInstallation"/>
160   <svrl:fired-rule context="core:CityObjectRelation"/>
161   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
162   <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
163                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][4]/*:
BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
://www.opengis.net/citygml/building/3.0'][1]">
164     <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
xml'</svrl:text>
165   </svrl:failed-assert>
166   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
$codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
//gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
```

```
167                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][4]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
168       <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
169     </svrl:failed-assert>
170     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
       and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
       $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
       substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
       [gml:identifier = $codeListValue])"
171                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][4]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
172       <svrl:text>Code list value shall exist</svrl:text>
173     </svrl:failed-assert>
174     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
175     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
176     <svrl:fired-rule context="bldg:BuildingInstallation"/>
177     <svrl:fired-rule context="core:CityObjectRelation"/>
178     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
179     <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
180                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][5]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
181       <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
       xml'</svrl:text>
182     </svrl:failed-assert>
183     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
       $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
       ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
       //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
184                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][5]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
185       <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
186     </svrl:failed-assert>
187     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
       and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
       $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
       substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
       [gml:identifier = $codeListValue])"
188                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][5]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
189       <svrl:text>Code list value shall exist</svrl:text>
190     </svrl:failed-assert>
191     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
192     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
193     <svrl:fired-rule context="bldg:BuildingInstallation"/>
194     <svrl:fired-rule context="core:CityObjectRelation"/>
195     <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
196     <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
197                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
       building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][6]/*:
       BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
       ://www.opengis.net/citygml/building/3.0'][1]">
198       <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
       xml'</svrl:text>
199     </svrl:failed-assert>
200     <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
       SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
       $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
       ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
       //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
201                     location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
       namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
```

```
        building/3.0']/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][6]/*:
        BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
        ://www.opengis.net/citygml/building/3.0'][1]">
202     <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
203   </svrl:failed-assert>
204   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
        $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])"
205                   location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
        namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
        building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][6]/*:
        BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
        ://www.opengis.net/citygml/building/3.0'][1]">
206     <svrl:text>Code list value shall exist</svrl:text>
207   </svrl:failed-assert>
208   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
209   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
210   <svrl:fired-rule context="bldg:BuildingInstallation"/>
211   <svrl:fired-rule context="core:CityObjectRelation"/>
212   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
213   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
        $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
        substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
        [gml:identifier = $codeListValue])"
214                   location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
        namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
        building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][7]/*:
        BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
        ://www.opengis.net/citygml/building/3.0'][1]">
215     <svrl:text>Code list value shall exist</svrl:text>
216   </svrl:failed-assert>
217   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
218   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
219   <svrl:fired-rule context="bldg:BuildingInstallation"/>
220   <svrl:fired-rule context="core:CityObjectRelation"/>
221   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
222   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
223   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
224   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
225   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
226   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
227   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
228   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
229   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
230   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
231   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
232   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
233   <svrl:fired-rule context="clrnceSpce:SurfaceProperties"/>
234   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:class"/>
235   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:function"/>
236   <svrl:fired-rule context="clrnceSpce:SurfaceProperties/clrnceSpce:usage"/>
237   <svrl:fired-rule context="bldg:BuildingInstallation/bldg:class"/>
238   <svrl:failed-assert test="if (not(@codeSpace)) then true() else @codeSpace='file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml'"
239                   location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
        namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
        building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][8]/*:
        BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
        ://www.opengis.net/citygml/building/3.0'][1]">
240     <svrl:text>Code space, if set, shall be 'file:///C:/Users/felix/Documents/SaxonXLSTProcessor/exampleCodeList_GML32.
        xml'</svrl:text>
241   </svrl:failed-assert>
242   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml' satisfies (not(contains($codeListUrl, '#')) and doc-available(
        $codeListUrl) and doc($codeListUrl)/gml:Dictionary) or (contains($codeListUrl, '#') and doc-available(substring-before
        ($codeListUrl,'#')) and boolean(for $codelistDoc in doc(substring-before($codeListUrl,'#')) return exists($codelistDoc
        //gml:Dictionary[@gml:id = substring-after($codeListUrl,'#')])))"
243                   location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
        namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
        building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][8]/*:
        BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
        ://www.opengis.net/citygml/building/3.0'][1]">
244     <svrl:text>Code list dictionary shall be represented using a GML 3.2 Dictionary</svrl:text>
245   </svrl:failed-assert>
246   <svrl:failed-assert test="every $codeListUrl in if (@codeSpace) then @codeSpace else 'file:///C:/Users/felix/Documents/
        SaxonXLSTProcessor/exampleCodeList_GML32.xml', $codeListValue in ./text() satisfies(not(contains($codeListUrl, '#'))
        and doc-available($codeListUrl) and doc($codeListUrl)/*/gml:dictionaryEntry/gml:Definition[gml:identifier =
```

```
      $codeListValue]) or (contains($codeListUrl, '#') and doc-available(substring-before($codeListUrl,'#')) and doc(
      substring-before($codeListUrl,'#'))//*[@gml:id = substring-after($codeListUrl,'#')]/gml:dictionaryEntry/gml:Definition
      [gml:identifier = $codeListValue])"
247                   location="/*:CityModel[namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:cityObjectMember[
      namespace-uri()='http://www.opengis.net/citygml/3.0'][1]/*:Building[namespace-uri()='http://www.opengis.net/citygml/
      building/3.0'][1]/*:buildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][8]/*:
      BuildingInstallation[namespace-uri()='http://www.opengis.net/citygml/building/3.0'][1]/*:class[namespace-uri()='http
      ://www.opengis.net/citygml/building/3.0'][1]">
248       <svrl:text>Code list value shall exist</svrl:text>
249    </svrl:failed-assert>
250    <svrl:fired-rule context="bldg:BuildingInstallation/bldg:function"/>
251    <svrl:fired-rule context="bldg:BuildingInstallation/bldg:usage"/>
252 </svrl:schematron-output>
```

Listing B.5: validationReportCityGML3.0.xml

C SiteplanADE Objektartenkatalog

Feature Catalogue Siteplan ADE

Inhaltsverzeichnis

1 Feature Catalogue Siteplan ADE 1
1.1 Application schema: Siteplan 1
1.2 Package: LandUsePlanning 1
1.3 Package: LandUsePlanningLADM 3
1.3.1 SP_PlanningUnit 3
1.4 Package: LandUsePlanningINSPIRE 4
1.4.1 BackgroundMapValue 5
1.4.2 DimensioningIndicationCharacterValue 5
1.4.3 DimensioningIndicationIntegerValue 6
1.4.4 DimensioningIndicationMeasureValue 6
1.4.5 DimensioningIndicationRealValue 7
1.4.6 DimensioningIndicationValue 7
1.4.7 DocumentCitation 8
1.4.8 HILUCSPercentage 9
1.4.9 HILUCSPresence 10
1.4.10 LegislationCitation 11
1.4.11 OfficialDocumentation 13
1.4.12 OfficialJournalInfomation 14
1.4.13 SpecificPercentage 15
1.4.14 SpecificPresence 16
1.4.15 SupplementaryRegulation 16
1.4.16 ZoningElement 21
1.5 Package: LandUsePlanningXPlanung 24
1.5.1 BP_AbgrabungsFlaeche 36
1.5.2 BP_AbstandsFlaeche 37
1.5.3 BP_AbstandsMass 37
1.5.4 BP_AbweichungVonBaugrenze 39
1.5.5 BP_AbweichungVonUeberbaubererGrundstuecksFlaeche 39
1.5.6 BP_AnpflanzungBindungErhaltung 39
1.5.7 BP_AufschuettungsFlaeche 42
1.5.8 BP_AusgleichsFlaeche 42
1.5.9 BP_AusgleichsMassnahme 44
1.5.10 BP_BauGrenze 45
1.5.11 BP_BauLinie 46
1.5.12 BP_BaugebietsTeilFlaeche 47
1.5.13 BP_BereichOhneEinAusfahrtLinie 53
1.5.14 BP_BesondererNutzungszweckFlaeche 54

1.5.15 BP_BodenschaetzeFlaeche 55
1.5.16 BP_Dachgestaltung 56
1.5.17 BP_EinfahrtPunkt 58
1.5.18 BP_EinfahrtsbereichLinie 58
1.5.19 BP_Eingriffsbereich 59
1.5.20 BP_EmissionskontingentLaerm 59
1.5.21 BP_EmissionskontingentLaermGebiet 60
1.5.22 BP_ErhaltungsBereichFlaeche 61
1.5.23 BP_FestsetzungNachLandesrecht 61
1.5.24 BP_FestsetzungenBaugebiet 62
1.5.25 BP_FirstRichtungsLinie 71
1.5.26 BP_FoerderungsFlaeche 71
1.5.27 BP_FreiFlaeche 72
1.5.28 BP_GemeinbedarfsFlaeche 72
1.5.29 BP_GemeinschaftsanlagenFlaeche 76
1.5.30 BP_GemeinschaftsanlagenZuordnung 77
1.5.31 BP_GestaltungBaugebiet 78
1.5.32 BP_GewaesserFlaeche 81
1.5.33 BP_GruenFlaeche 82
1.5.34 BP_HoehenMass 84
1.5.35 BP_Immissionsschutz 85
1.5.36 BP_KennzeichnungsFlaeche 87
1.5.37 BP_KleintierhaltungFlaeche 88
1.5.38 BP_Landwirtschaft 88
1.5.39 BP_LandwirtschaftsFlaeche 89
1.5.40 BP_NebenanlagenAusschlussFlaeche 90
1.5.41 BP_NebenanlagenFlaeche 91
1.5.42 BP_NichtUeberbaubareGrundstuecksflaeche 92
1.5.43 BP_NutzungsartenGrenze 93
1.5.44 BP_Objekt 94
1.5.45 BP_PersGruppenBestimmteFlaeche 97
1.5.46 BP_RegelungVergnuegungsstaetten 98
1.5.47 BP_RekultivierungsFlaeche 98
1.5.48 BP_Richtungssektor 99
1.5.49 BP_RichtungssektorGrenze 100
1.5.50 BP_SchutzPflegeEntwicklungsFlaeche 100
1.5.51 BP_SchutzPflegeEntwicklungsMassnahme 101
1.5.52 BP_Sichtflaeche 103

1.5.53 BP_SpezielleBauweise 103
1.5.54 BP_SpielSportanlagenFlaeche....105
1.5.55 BP_StrassenVerkehrsFlaeche 106
1.5.56 BP_Strassenbegrenzungslinie 107
1.5.57 BP_Strassenkoerper 107
1.5.58 BP_TechnischeMassnahmenFlaeche 108
1.5.59 BP_TextlicheFestsetzungsFlaeche 109
1.5.60 BP_UeberbaubareGrundstuecksFlaeche 109
1.5.61 BP_UnverbindlicheVormerkung 113
1.5.62 BP_VerEntsorgung 113
1.5.63 BP_Veraenderungssperre....117
1.5.64 BP_VerkehrsflaecheBesondererZweckbestimmung....118
1.5.65 BP_WaldFlaeche 120
1.5.66 BP_WasserwirtschaftsFlaeche....122
1.5.67 BP_ZusaetzlicheFestsetzungen 123
1.5.68 BP_ZusatzkontingentLaerm 125
1.5.69 BP_ZusatzkontingentLaermFlaeche....126
1.5.70 FP_Objekt....127
1.5.71 RP_Objekt....127
1.5.72 RP_Wasserschutz....129
1.5.73 SO_Denkmalschutzrecht....130
1.5.74 SO_Gebiet 132
1.5.75 SO_Gewaesser....135
1.5.76 SO_Objekt 136
1.5.77 SO_SonstigesRecht....137
1.5.78 XP_Hoehenangabe....138
1.5.79 XP_Objekt....141
1.5.80 XP_SPEMassnahmenDaten 143
1.5.81 XP_Wirksamkeitsbedingung....145
1.6 Package: LandAdministrationCadastre....145
1.7 Package: LandAdministrationCadastre_LADM....146
1.7.1 Fraction....147
1.7.2 LA_AdministrativeSource 147
1.7.3 LA_BAUnit 148
1.7.4 LA_BoundaryFaceString 149
1.7.5 LA_GroupParty 150
1.7.6 LA_Mortgage....150
1.7.7 LA_Party 151

1.7.8 LA_Point 153
1.7.9 LA_RRR 153
1.7.10 LA_Responsibility 154
1.7.11 LA_Restriction 155
1.7.12 LA_Right 155
1.7.13 LA_SpatialUnitGroup 156
1.8 Package: LandAdministrationCadastre_INSPIRE 156
1.8.1 BasicPropertyUnit 157
1.8.2 CadastralBoundary 158
1.8.3 CadastralParcel 159
1.8.4 CadastralZoning 160
1.8.5 DerivedProfilePresenceInSoilBody 160
1.8.6 DerivedSoilProfile 161
1.8.7 FAOHorizonNotationType 161
1.8.8 OtherHorizonNotationType 165
1.8.9 ParticleSizeFractionType 166
1.8.10 ProfileElement 166
1.8.11 RangeType 168
1.8.12 SoilHorizon 169
1.8.13 SoilLayer 170
1.8.14 SoilProfile 172
1.8.15 SoilProperties 174
1.8.16 WRBQualifierGroupType 174
1.8.17 WRBSoilNameType 176
1.9 Package: AAA 177
1.9.1 AX_Aufnahmepunkt 185
1.9.2 AX_BesondereFlurstuecksgrenze 186
1.9.3 AX_Bodenschaetzung 187
1.9.4 AX_Buchungsblatt 196
1.9.5 AX_Buchungsblattbezrik_Schluessel 197
1.9.6 AX_Buchungstelle 198
1.9.7 AX_Bundesland_Schluessel 203
1.9.8 AX_DQFestpunkt 203
1.9.9 AX_DQHoehenfestpunkt 205
1.9.10 AX_Dienststelle_Schluessel 207
1.9.11 AX_Festpunkt 207
1.9.12 AX_Flurstueck 222
1.9.13 AX_Flurstuecksnummer 227

1.9.14 AX_Gemarkung....228
1.9.15 AX_Gemarkung_Schluessel....229
1.9.16 AX_Gemeindekennzeichen....230
1.9.17 AX_Grenzpunkt....231
1.9.18 AX_Hoehenfestpunkt....247
1.9.19 AX_Klassifikation_Lagefestpunkt....248
1.9.20 AX_LagebezeichnungMitHausnummer....250
1.9.21 AX_LagebezeichnungOhneHausnummer....251
1.9.22 AX_Lagefestpunkt....251
1.9.23 AX_Nammensnummer....253
1.9.24 AX_Netzpunkt....259
1.9.25 AX_Person....272
1.9.26 AX_PersonenGruppe....275
1.9.27 AX_Pfeilerhoehe_Lagefestpunkt....275
1.9.28 AX_Punktstabilitaet_Hoehenfestpunkt....276
1.9.29 AX_Sicherungspunkt....280
1.9.30 AX_SonstigeEigenschaften_Flurstueck....281
1.9.31 AX_SonstigerVermessungspunkt....282
1.9.32 AX_Vertretung....283
1.10 Package: ConstructionLaw....284
1.10.1 AbstractConstructionLawObject....284
1.10.2 BuildingContractor....284
1.10.3 BuildingPermitParty....285
1.10.4 ClearanceSpace....286
1.10.5 ConstructionLawProperties....287
1.10.6 ConstructionSupervision....287
1.10.7 Editor....288
1.10.8 Municipality....288
1.10.9 Neighbour....289
1.11 Package: RiskAssessment....289
1.11.1 AbstractHazardArea....290
1.11.2 AbstractRiskZone....291
1.11.3 CadastralMap....292
1.11.4 ExposedElementClassification....292
1.11.5 ExposedElementProperties....293
1.11.6 HazardArea....293
1.11.7 LevelOrIntensity....294
1.11.8 LikelihoodOfOccurrence....295

1.11.9 NaturalHazardClassification 296
1.11.10 ObservedEvent 296
1.11.11 PhysicalProperties 297
1.11.12 QuantitativeLikelihood 298
1.11.13 RiskZone 299
1.11.14 VulnerabilityAssessment 300
1.12 Package: Building 301
1.12.1 MeasureOfStructuralUse 301
1.12.2 SurfaceProperties 303
1.13 Package: RegulationConflict 304
1.13.1 Conflict 305
1.13.2 ConstructionLawConflict 307
1.13.3 ExaminationNote 307
1.13.4 LandUseRegulationConflict 308
1.14 Package: Core 308
1.14.1 ImprovedGeometry 309
1.14.2 ObjectMetaData 310
1.14.3 SiteplanMetaData 312
1.15 Package: ThematicGrouping 314
1.15.1 ExisitingConceptualObjects 314
1.15.2 ExisitingObjects 315
1.15.3 ExistingPrivateLawObjects 315
1.15.4 ExistingPublicLawObjects 315
1.15.5 ExistingTopographicObjects 315
1.15.6 PlannedConceptualObjects 316
1.15.7 PlannedObjects 316
1.15.8 PlannedPrivateLawObjects 316
1.15.9 PlannedPublicLawObjects 316
1.15.10 PlannedTopographicObjects 316
1.15.11 PrivateSpaceObjects 317
1.15.12 PublicAuthority 317
1.15.13 PublicSpaceObjects 317
1.15.14 SpatialPlan 317

1 Feature Catalogue Siteplan ADE

Version:

1.0.0

Date:

2021-01-18T18:04:52.3306994+01:00

Scope:

Digital 3D site plan.

Responsible organisation:

Technische Universität München

1.1 Application schema: Siteplan

Definition:

The Siteplan application schema provides classes to model a 3D representation of a site plan.

Sub-package:

Package: Building

Package: ConstructionLaw

Package: Core

Package: LandAdministrationCadastre

Package: LandUsePlanning

Package: RegulationConflict

Package: RiskAssessment

Package: ThematicGrouping

1.2 Package: LandUsePlanning

Definition:

The LandUsePlanning package provides classes on different levels, i.e. from international to national scope, to represent objects of the spatial plans.

Sub-package:

Package: LandUsePlanningINSPIRE

Package: LandUsePlanningLADM

Package: LandUsePlanningXPlanung

Parent package:

Application schema: Siteplan

Diagram(s):

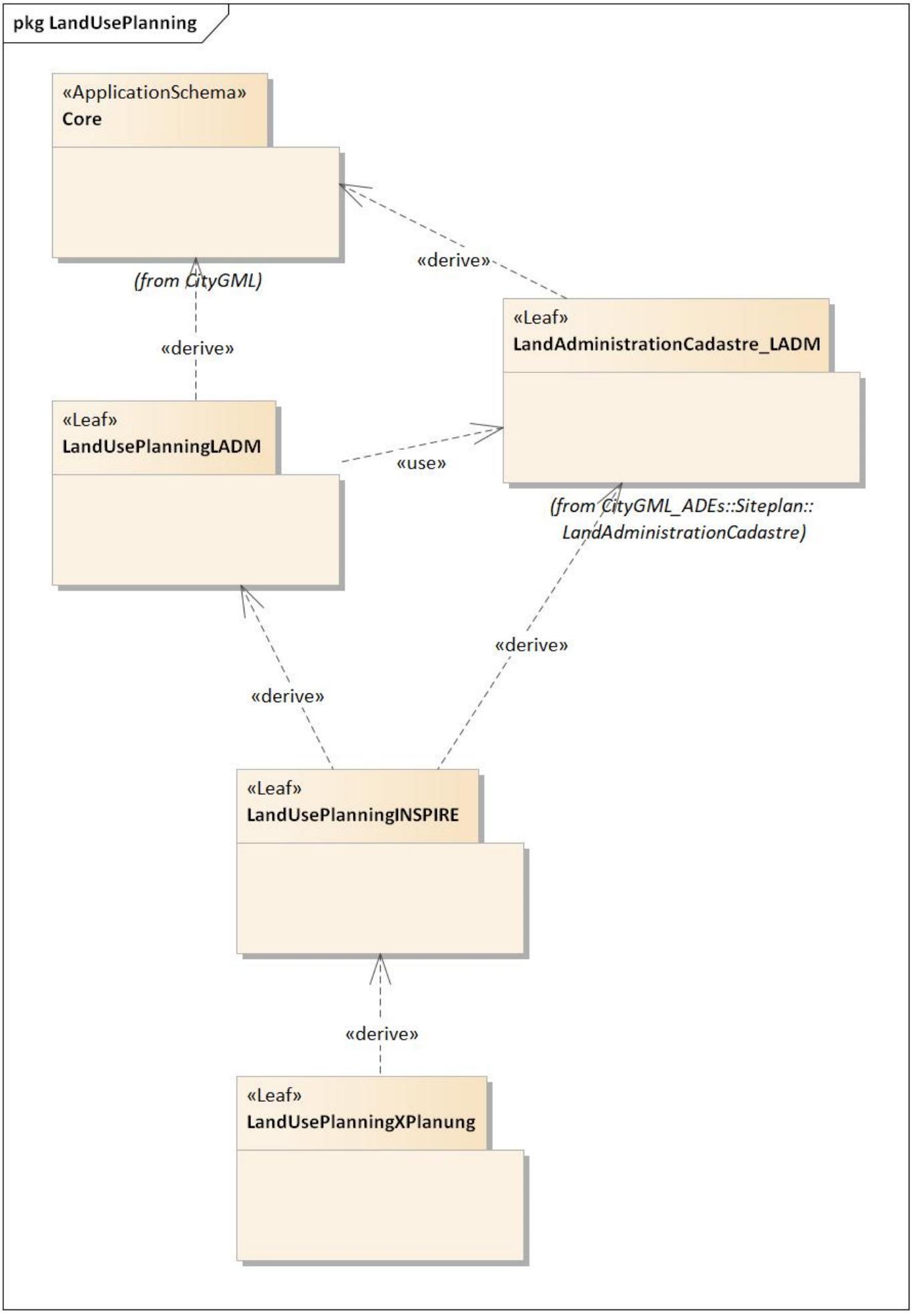

pkg LandUsePlanning
«ApplicationSchema»
Core
(from CityGML)
«derive»
«derive»
«Leaf»
LandAdministrationCadastre_LADM
«Leaf»
LandUsePlanningLADM
«use»
(from CityGML_ADEs::Siteplan::
LandAdministrationCadastre)
«derive»
«derive»
«Leaf»
LandUsePlanningINSPIRE
«derive»
«Leaf»
LandUsePlanningXPlanung

Diagram 1 - LandUsePlanning

1.3 Package: LandUsePlanningLADM

Parent package:

Package: LandUsePlanning

Diagram(s):

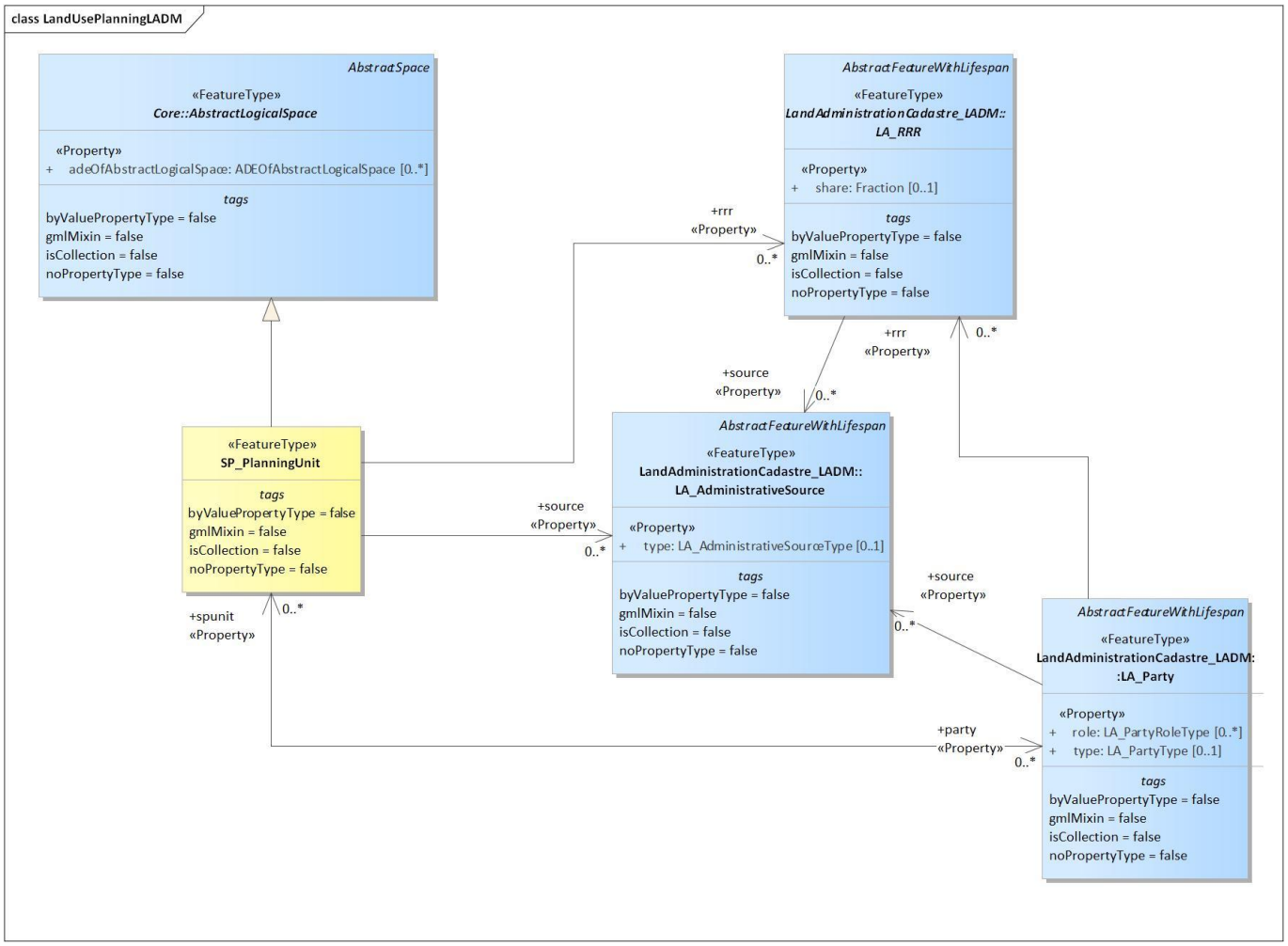


Diagram 2 - LandUsePlanningLADM

1.3.1 SP_PlanningUnit

SP_PlanningUnit	
Definition:	Registers a zoning unit which may contain RRRs.
Subtype of:	AbstractLogicalSpace
Supertype of:	SupplementaryRegulation
	ZoningElement
Type:	Feature type
Association role	
Name:	party

Voidable:	false
Multiplicity:	0..*
Value type:	LA_Party (feature type)
Association role	
Name:	rrr
Voidable:	false
Multiplicity:	0..*
Value type:	LA_RRR (feature type)
Association role	
Name:	source
Voidable:	false
Multiplicity:	0..*
Value type:	LA_AdministrativeSource (feature type)

1.4 Package: LandUsePlanningINSPIRE

Parent package:

Package: LandUsePlanning

Diagram(s):

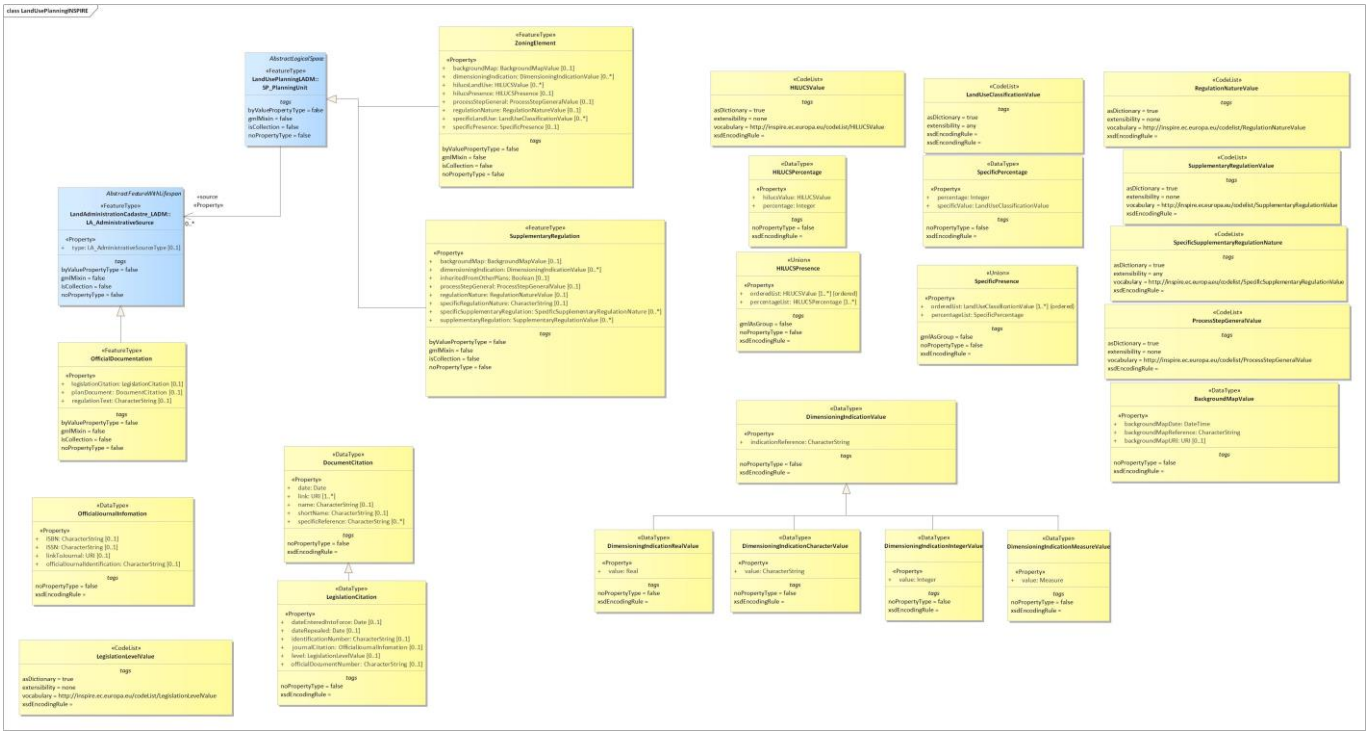


Diagram 3 - LandUsePlanningINSPIRE

1.4.1 BackgroundMapValue

BackgroundMapValue	
Definition:	From the INSPIRE specification: Information regarding the map that has been used as a background in the definition of a spatial plan, a zoning element or a supplementary regulation.
Type:	Data type
Attribute:	
Name:	backgroundMapDate
Definition:	From the INSPIRE specification: Date of the background map used.
Voidable:	false
Multiplicity:	1
Value type:	DateTime
Attribute:	
Name:	backgroundMapReference
Definition:	From the INSPIRE specification: Reference to the background map that has been used.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	backgroundMapURI
Definition:	From the INSPIRE specification: URI referring to service that provides background map.
Voidable:	false
Multiplicity:	0..1
Value type:	URI

1.4.2 DimensioningIndicationCharacterValue

DimensioningIndicationCharacterValue	
Definition:	From the INSPIRE specification: Dimensioning indication whose value is of type CharacterString.

Subtype of:	DimensioningIndicationValue
Type:	Data type
Attribute:	
Name:	value
Definition:	From the INSPIRE specification: Dimensioning indication whose value is a text.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.4.3 DimensioningIndicationIntegerValue

DimensioningIndicationIntegerValue	
Definition:	From the INSPIRE specification: Dimensioning indication whose value is of type integer.
Subtype of:	DimensioningIndicationValue
Type:	Data type
Attribute:	
Name:	value
Definition:	From the INSPIRE specification: Dimensioning indication whose value is a integer number.
Voidable:	false
Multiplicity:	1
Value type:	Integer

1.4.4 DimensioningIndicationMeasureValue

DimensioningIndicationMeasureValue	
Definition:	From the INSPIRE specification: Dimensioning indication whose value is a measure.
Subtype of:	DimensioningIndicationValue
Type:	Data type
Attribute:	
Name:	value
Definition:	From the INSPIRE specification:

Voidable:	false
Multiplicity:	1
Value type:	Measure

1.4.5 DimensioningIndicationRealValue

DimensioningIndicationRealValue	
Definition:	From the INSPIRE specification: Dimensioning indication whose value is a floating point number.
Subtype of:	DimensioningIndicationValue
Type:	Data type
Attribute:	
Name:	value
Definition:	From the INSPIRE specification:
Voidable:	false
Multiplicity:	1
Value type:	Real

1.4.6 DimensioningIndicationValue

DimensioningIndicationValue	
Definition:	From the INSPIRE specification: Specifications about the dimensioning of the urban developments.
Supertype of:	DimensioningIndicationCharacterValue DimensioningIndicationIntegerValue DimensioningIndicationMeasureValue DimensioningIndicationRealValue
Type:	Data type
Attribute:	
Name:	indicationReference
Definition:	From the INSPIRE specification: Description of the dimension indication.
Voidable:	false
Multiplicity:	1

Value type:	CharacterString

1.4.7 DocumentCitation

DocumentCitation	
Definition:	From the INSPIRE specification: Citation for the purposes of unambiguously referencing a document.
Supertype of:	LegislationCitation
Type:	Data type
Attribute:	
Name:	date
Definition:	From the INSPIRE specification: Date of creation, publication or revision of the document.
Voidable:	false
Multiplicity:	1
Value type:	Date
Attribute:	
Name:	link
Definition:	From the INSPIRE specification: Link to an online version of the document.
Voidable:	false
Multiplicity:	1..*
Value type:	URI
Attribute:	
Name:	name
Definition:	From the INSPIRE specification: Name of the document. NOTE For legal documents, this should be the official name assigned to the legislative instrument. EXAMPLE The official legal name for the INSPIRE Directive is "Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE)"
Voidable:	false

Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	shortName
Definition:	From the INSPIRE specification: Short name or alternative title of the document. NOTE For legal documents, this should be a short name or alternative title commonly used to identify the legislation.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	specificReference
Definition:	From the INSPIRE specification: Reference to a specific part of the document. EXAMPLE For legal documents, this attribute can contain a reference to article(s) that specify a specific requirement or obligation.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString

1.4.8 HILUCSPercentage

HILUCSPercentage	
Definition:	From the INSPIRE specification: Percentage of land use object that is covered by this HILUCS presence. NOTE1:The percentage is according to the socio-economic or functional importance of the use. NOTE2:examples are provided in the narrative description part of the data specification.
Type:	Data type
Attribute:	
Name:	hilucsValue
Definition:	From the INSPIRE specification:

	HILUCS category for this HILUCS percentage.
Voidable:	false
Multiplicity:	1
Value type:	HILUCSValue (code list)
Attribute:	
Name:	percentage
Definition:	From the INSPIRE specification: Percentage of land use object that is covered by this specific presence. NOTE The percentage is according to the socio-economic importance of the use. In section 5.2.1.1.2. examples of the use of percentages are given.
Voidable:	false
Multiplicity:	1
Value type:	Integer

1.4.9 HILUCSPresence

HILUCSPresence	
Definition:	From the INSPIRE specification: Presence of one or several HILUCS values in an area, indicated either as the percentage covered for each value or as the values listed in their order of importance. The HILUCS presence data type enables the provision of information on land uses inside one land use object in order to collect more than one land use existence perfectly identifiable by importance order or percentages. NOTE 1: The order of land use value presence without percentages enable providing an order of dominance/importance of each land use present in the land use object NOTE 2: The sum of the percentages can be below 100%, or above. The order is provided according to the respective importance when the percentages are not known.
Type:	Union type
Attribute:	
Name:	orderedList
Voidable:	false

Multiplicity:	1..*
Value type:	HILUCSValue (code list)
Attribute:	
Name:	percentageList
Voidable:	false
Multiplicity:	1..*
Value type:	HILUCSPercentage (data type)

1.4.10 LegislationCitation

LegislationCitation	
Definition:	From the INSPIRE specification: Citation for the purposes of unambiguously referencing a legal act or a specific part of a legal act.
Subtype of:	DocumentCitation
Type:	Data type
Attribute:	
Name:	dateEnteredIntoForce
Definition:	From the INSPIRE specification: Date the legislative instrument entered into force.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	dateRepealed
Definition:	From the INSPIRE specification: Date the legislative instrument was repealed.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	identificationNumber
Definition:	From the INSPIRE specification:

	EXAMPLE 1: 2007/2/EC is the identification number for the INSPIRE Directive EXAMPLE 2: 2008/50/EC is the identification number for the CAFE Directive EXAMPLE 3: 2000/60/EC is the identification number for the Water Framework Directive
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	journalCitation
Definition:	From the INSPIRE specification: Citation of the official journal in which the legislation is published.
Voidable:	false
Multiplicity:	0..1
Value type:	OfficialJournalInfomation (data type)
Attribute:	
Name:	level
Definition:	From the INSPIRE specification: The level at which the legislative instrument is adopted.
Voidable:	false
Multiplicity:	0..1
Value type:	LegislationLevelValue (code list)
Attribute:	
Name:	officialDocumentNumber
Definition:	From the INSPIRE specification: Official document number used to uniquely identify the legislative instrument. NOTE: An official document number may be assigned to uniquely identify the legislative instrument. EXAMPLE: CELEX Number used to uniquely identify European Union Legislation
Voidable:	false
Multiplicity:	0..1

Value type:	CharacterString

1.4.11 OfficialDocumentation

OfficialDocumentation	
Definition:	From the INSPIRE specification: The official documentation that composes the spatial plan; it may be composed of, the applicable legislation, the regulations, cartographic elements, descriptive elements that may be associated with the complete spatial plan, a zoning element or a supplementary regulation . In some Member States the actual textual regulation will be part of the data set (and can be put in the regulationText attribute), in other Member States the text will not be part of the data set and will be referenced via a reference to a document or a legal act.
Subtype of:	LA_AdministrativeSource
Type:	Feature type
Attribute:	
Name:	legislationCitation
Definition:	From the INSPIRE specification: Reference to the document that contains the text of the regulation.
Voidable:	false
Multiplicity:	0..1
Value type:	LegislationCitation (data type)
Attribute:	
Name:	planDocument
Definition:	From the INSPIRE specification: Citation of scanned plans and structural drawings which may sometimes be geo-referenced or not,. E.g. raster images, vector drawings or scanned text.
Voidable:	false
Multiplicity:	0..1
Value type:	DocumentCitation (data type)
Attribute:	
Name:	regulationText
Definition:	From the INSPIRE specification:

	Text of the regulation.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.4.12 OfficialJournalInfomation

OfficialJournalInfomation	
Definition:	From the INSPIRE specification: Full citation of the location of the legislative instrument within the official journal.
Type:	Data type
Attribute:	
Name:	ISBN
Definition:	From the INSPIRE specification: International Standard Book Number (ISBN) is an nine-digit number that uniquely identifies the book in which the legislative instrument was published.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	ISSN
Definition:	From the INSPIRE specification: The International Standard Serial Number (ISSN) is an eight-digit number that identifies the periodical publication in which the legislative instrument was published. NOTE: Periodical publilcations are issued in successive parts, usually having numerical or chronological designations and required that each serial publication can be uniquely identified. EXAMPLE: OJ Series in which INSPIRE Directive is published has been assigned the ISSN: 1725-2555
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	

Name:	linkToJournal
Definition:	From the INSPIRE specification: Link to an online version of the official journal.
Voidable:	false
Multiplicity:	0..1
Value type:	URI
Attribute:	
Name:	officialJournalIdentification
Definition:	From the INSPIRE specification: Reference to the location within the official journal within which the legislative instrument was published. This reference shall be comprised of three parts: - the title of the official journal - the volume and/or series number - Page number(s) EXAMPLE: Official Journal of European Union (OJEU), L108, Volume 50, 1-14
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.4.13 SpecificPercentage

SpecificPercentage	
Definition:	From the INSPIRE specification: Percentage of land use object that is covered by this specific presence.
Type:	Data type
Attribute:	
Name:	percentage
Definition:	From the INSPIRE specification: Percentage of land use object that is covered by a specific presence. NOTE The percentage is according to the socio-economic importance of the use. In section 5.2.1.1.2. examples of the use of percentages are given.

Voidable:	false
Multiplicity:	1
Value type:	Integer
Attribute:	
Name:	specificValue
Definition:	From the INSPIRE specification: Specific value category for this specific percentage.
Voidable:	false
Multiplicity:	1
Value type:	LandUseClassificationValue (code list)

1.4.14 SpecificPresence

SpecificPresence	
Definition:	From the INSPIRE specification: Presence of one or several land use classification values in an area according to the code list provided by the data provider, indicated either as the percentage covered for each value or as the values listed in their order of importance.
Type:	Union type
Attribute:	
Name:	orderedList
Voidable:	false
Multiplicity:	1..*
Value type:	LandUseClassificationValue (code list)
Attribute:	
Name:	percentageList
Voidable:	false
Multiplicity:	1
Value type:	SpecificPercentage (data type)

1.4.15 SupplementaryRegulation

SupplementaryRegulation	
Definition:	From the INSPIRE specification:

	A spatial object (point, line or polygon) of a spatial plan that provides supplementary information and/or limitation of the use of land/water necessary for spatial planning reasons or to formalise external rules defined in legal text. NOTE the supplementary regulations affects all land use that overlap with the geometry EXAMPLE an air field generates restriction in its surroundings regarding aircraft landing, radar and telecommunication devices. It is the buffer around these artefacts that generates the supplementary regulation on the Land Use
Subtype of:	SP_PlanningUnit
Supertype of:	BP_AbstandsFlaeche BP_AbstandsMass BP_AbweichungVonBaugrenze BP_AbweichungVonUeberbaubererGrundstuecksFlaeche BP_AnpflanzungBindungErhaltung BP_AusgleichsFlaeche BP_AusgleichsMassnahme BP_BauGrenze BP_BauLinie BP_BereichOhneEinAusfahrtLinie BP_EinfahrtPunkt BP_EinfahrtsbereichLinie BP_Eingriffsbereich BP_ErhaltungsBereichFlaeche BP_FirstRichtungsLinie BP_FoerderungsFlaeche BP_FreiFlaeche BP_GemeinschaftsanlagenFlaeche BP_GemeinschaftsanlagenZuordnung BP_GruenFlaeche BP_HoehenMass BP_Immissionsschutz BP_NebenanlagenAusschlussFlaeche

	BP_NebenanlagenFlaeche
	BP_NichtUeberbaubareGrundstuecksflaeche
	BP_NutzungsartenGrenze
	BP_PersGruppenBestimmteFlaeche
	BP_RegelungVergnuegungsstaetten
	BP_RichtungssektorGrenze
	BP_SchutzPflegeEntwicklungsFlaeche
	BP_SchutzPflegeEntwicklungsMassnahme
	BP_Sichtflaeche
	BP_SpezielleBauweise
	BP_StrassenVerkehrsFlaeche
	BP_Strassenbegrenzungslinie
	BP_Strassenkoerper
	BP_TechnischeMassnahmenFlaeche
	BP_TextlicheFestsetzungsFlaeche
	BP_UeberbaubareGrundstuecksFlaeche
	BP_UnverbindlicheVormerkung
	BP_Veraenderungssperre
	BP_VerkehrsflaecheBesondererZweckbestimmung
	BP_ZusatzkontingentLaerm
	BP_ZusatzkontingentLaermFlaeche
	SO_Denkmalschutzrecht
	SO_Gebiet
Type:	Feature type
Attribute:	
Name:	backgroundMap
Definition:	From the INSPIRE specification: Identification of the background map that has been used for constructing the supplementary regulation.
Voidable:	false
Multiplicity:	0..1
Value type:	BackgroundMapValue (data type)
Attribute:	

Name:	dimensioningIndication
Definition:	From the INSPIRE specification: Specifications about the dimensioning that are added to the dimensioning of the zoning elements that overlap the geometry of the supplementary regulation.
Voidable:	false
Multiplicity:	0..*
Value type:	DimensioningIndicationValue (data type)
Attribute:	
Name:	inheritedFromOtherPlans
Definition:	From the INSPIRE specification: Indication whether the supplementary regulation is inherited from another spatial plan.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean
Attribute:	
Name:	processStepGeneral
Definition:	From the INSPIRE specification: General indication of the step of the planning process that the supplementary regulation is undergoing. NOTE This enumeration contains values that are common to most planning systems.
Voidable:	false
Multiplicity:	0..1
Value type:	ProcessStepGeneralValue (code list)
Attribute:	
Name:	regulationNature
Definition:	From the INSPIRE specification: Legal nature of the land use regulation. NOTE Indicates whether the land use regulation is legally binding or not.
Voidable:	false
Multiplicity:	0..1
Value type:	RegulationNatureValue (code list)

Attribute:

Name:	specificRegulationNature
Definition:	From the INSPIRE specification: Legal nature of the land use regulation from a national perspective. On member state level the legal classification for plan regulations may be complex. A supplementary regulation may be principally "bindingOnlyForAuthorities" but according to national law this regulation have to be more specified to explain the specific degree of binding. (e.g. on state or regional planning level in Germany: principles of spatial planning, other spatial planning requirements or goals of spatial planning). NOTE This attribute complements the information provided by the attribute regulationNature.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	specificSupplementaryRegulation
Definition:	From the INSPIRE specification: Reference to a category of supplementary regulation provided in a specific nomenclature of supplementary regulations provided by the data provider.
Voidable:	false
Multiplicity:	0..*
Value type:	SpecificSupplementaryRegulationNature (code list)

Attribute:

Name:	supplementaryRegulation
Definition:	From the INSPIRE specification: Code of the supplementary regulation from the hierarchical supplementary regulation code list agreed at the European level.
Voidable:	false
Multiplicity:	0..*
Value type:	SupplementaryRegulationValue (code list)

1.4.16 ZoningElement

ZoningElement

Definition:	From the INSPIRE specification: A spatial object which is homogeneous regarding the permitted uses of land based on zoning which separate one set of land uses from another. Zoning elements refer to the regulation of the kinds of activities which will be acceptable on particular lots (such as open space, residential, agricultural, commercial or industrial). The intensity of use at which those activities can be performed (from low-density housing such as single family homes to high-density such as high-rise apartment buildings), the height of buildings, the amount of space that structures may occupy, the proportions of the types of space on a lot, such as how much landscaped space, impervious surface, traffic lanes, and parking may be provided.
Subtype of:	SP_PlanningUnit
Supertype of:	BP_AbgrabungsFlaeche BP_AufschuettungsFlaeche BP_BaugebietsTeilFlaeche BP_BesondererNutzungszweckFlaeche BP_BodenschaetzeFlaeche BP_FestsetzungNachLandesrecht BP_GemeinbedarfsFlaeche BP_GewaesserFlaeche BP_KennzeichnungsFlaeche BP_KleintierhaltungFlaeche BP_Landwirtschaft BP_LandwirtschaftsFlaeche BP_RekultivierungsFlaeche BP_SpielSportanlagenFlaeche BP_VerEntsorgung BP_WaldFlaeche BP_WasserwirtschaftsFlaeche RP_Wasserschutz SO_Gewaesser SO_SonstigesRecht

Type:	Feature type
Attribute:	
Name:	backgroundMap
Definition:	From the INSPIRE specification: Identification of the background map that has been used for constructing this zoning element.
Voidable:	false
Multiplicity:	0..1
Value type:	BackgroundMapValue (data type)
Attribute:	
Name:	dimensioningIndication
Definition:	From the INSPIRE specification: Specifications about the dimensioning of the urban developments.
Voidable:	false
Multiplicity:	0..*
Value type:	DimensioningIndicationValue (data type)
Attribute:	
Name:	hilucsLandUse
Definition:	From the INSPIRE specification: Land use HILUCS class that is dominant in this land use object.
Voidable:	false
Multiplicity:	0..*
Value type:	HILUCSValue (code list)
Attribute:	
Name:	hilucsPresence
Definition:	From the INSPIRE specification: Actual presence of a land use HILUCS category wihtin the object.
Voidable:	false
Multiplicity:	0..1
Value type:	HILUCSPresence (union data type)
Attribute:	

Name:	processStepGeneral
Definition:	From the INSPIRE specification: General indication of the step of the planning process that the zoning element is undergoing. NOTE This enumeration contains values that are common to most planning systems.
Voidable:	false
Multiplicity:	0..1
Value type:	ProcessStepGeneralValue (code list)
Attribute:	
Name:	regulationNature
Definition:	From the INSPIRE specification: Legal nature of the land use indication. NOTE Indicates whether the land use indication is legally binding or not.
Voidable:	false
Multiplicity:	0..1
Value type:	RegulationNatureValue (code list)
Attribute:	
Name:	specificLandUse
Definition:	From the INSPIRE specification: Land Use Category according to the nomenclature specific to this data set. Reference to an entry in the classfication that is part of the SpecificLandUseClassification.
Voidable:	false
Multiplicity:	0..*
Value type:	LandUseClassificationValue (code list)
Attribute:	
Name:	specificPresence
Definition:	From the INSPIRE specification: Actual presence of a land use category wihtin the object.
Voidable:	false
Multiplicity:	0..1

Value type:	SpecificPresence (union data type)

1.5 Package: LandUsePlanningXPlanung

Parent package:

Package: LandUsePlanning

Diagram(s):

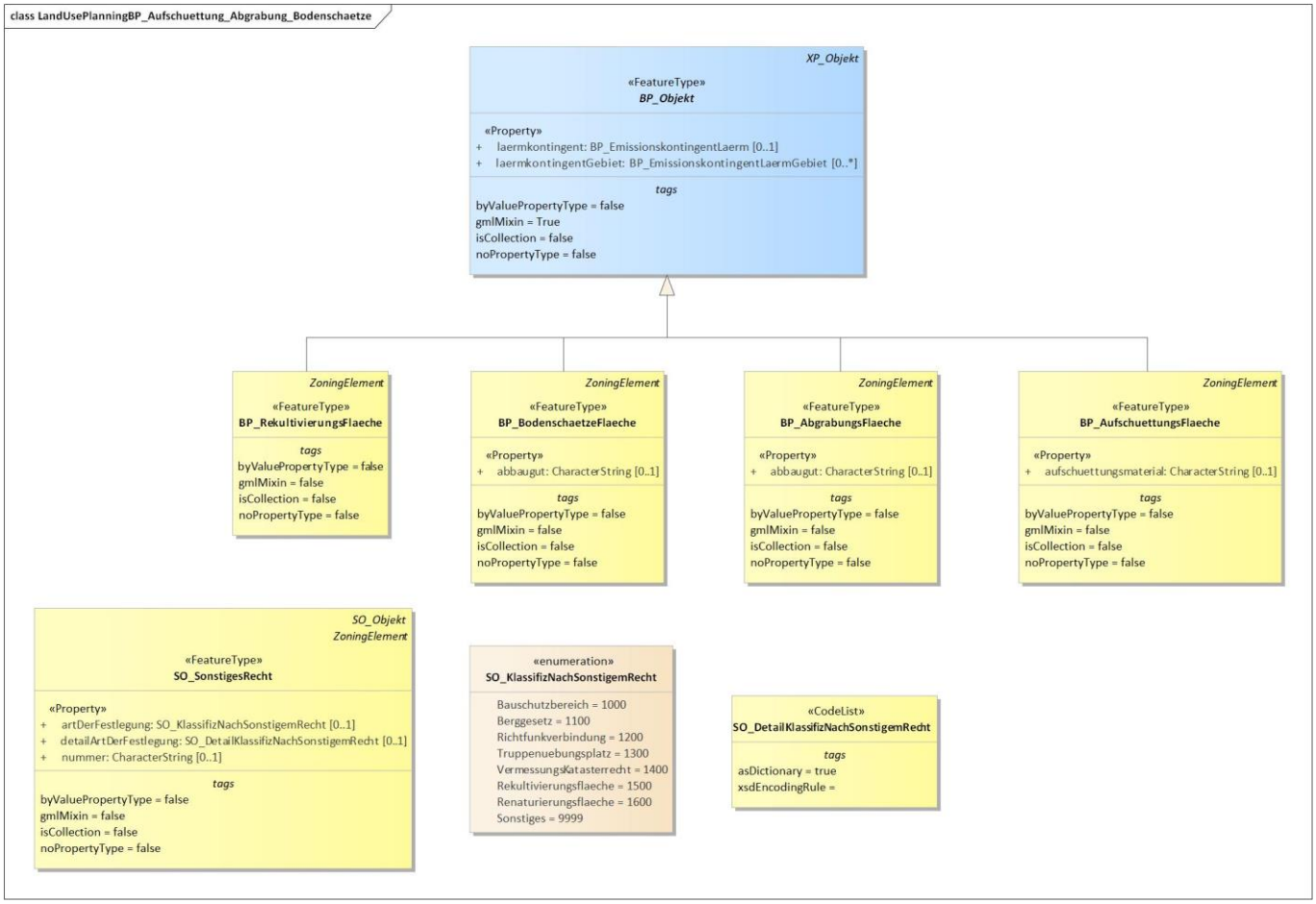


Diagram 4 - LandUsePlanningBP_Aufschuettung_Abgrabung_Bodenschaetze

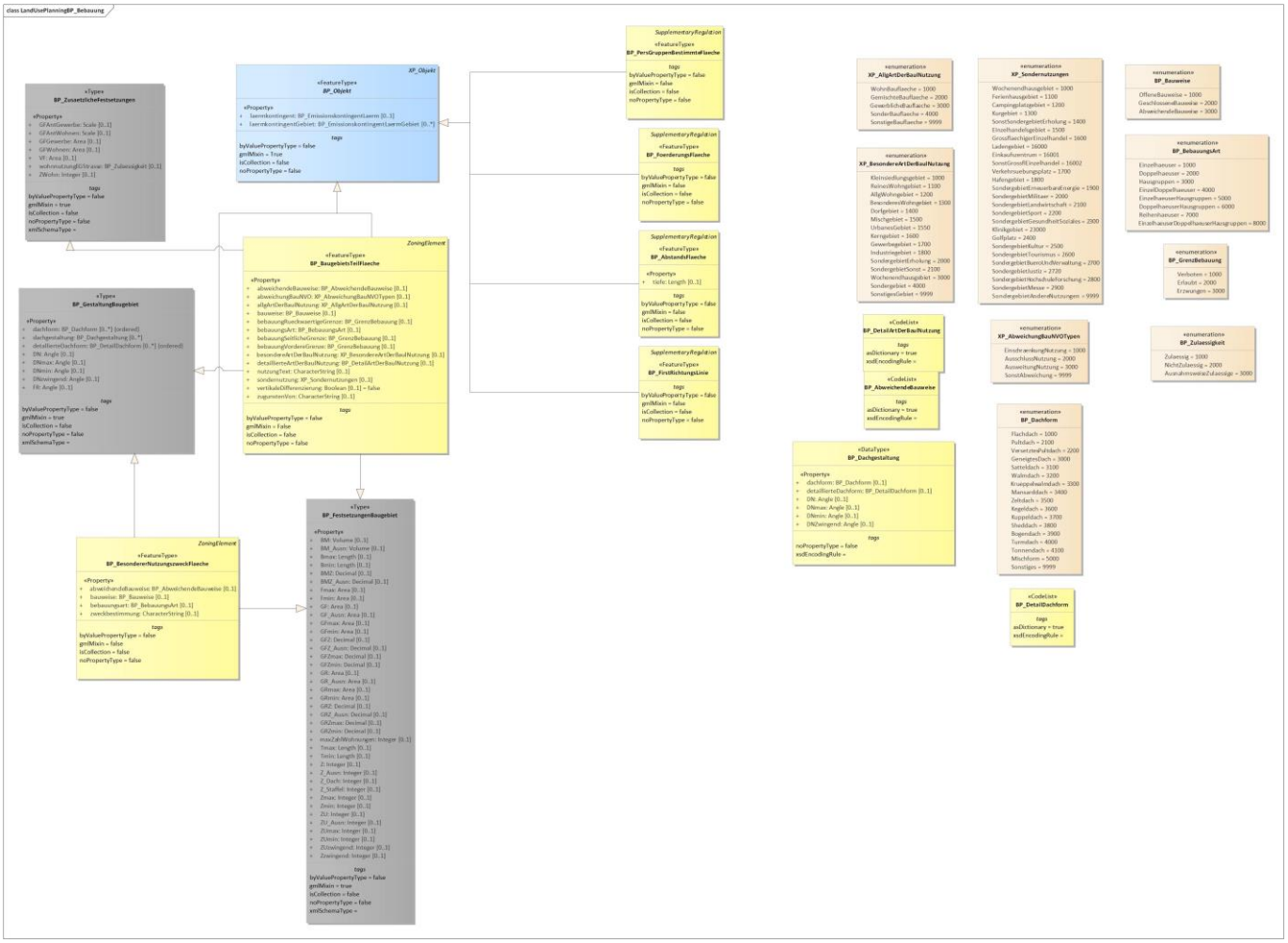


Diagram 5 - LandUsePlanningBP_Bebauung

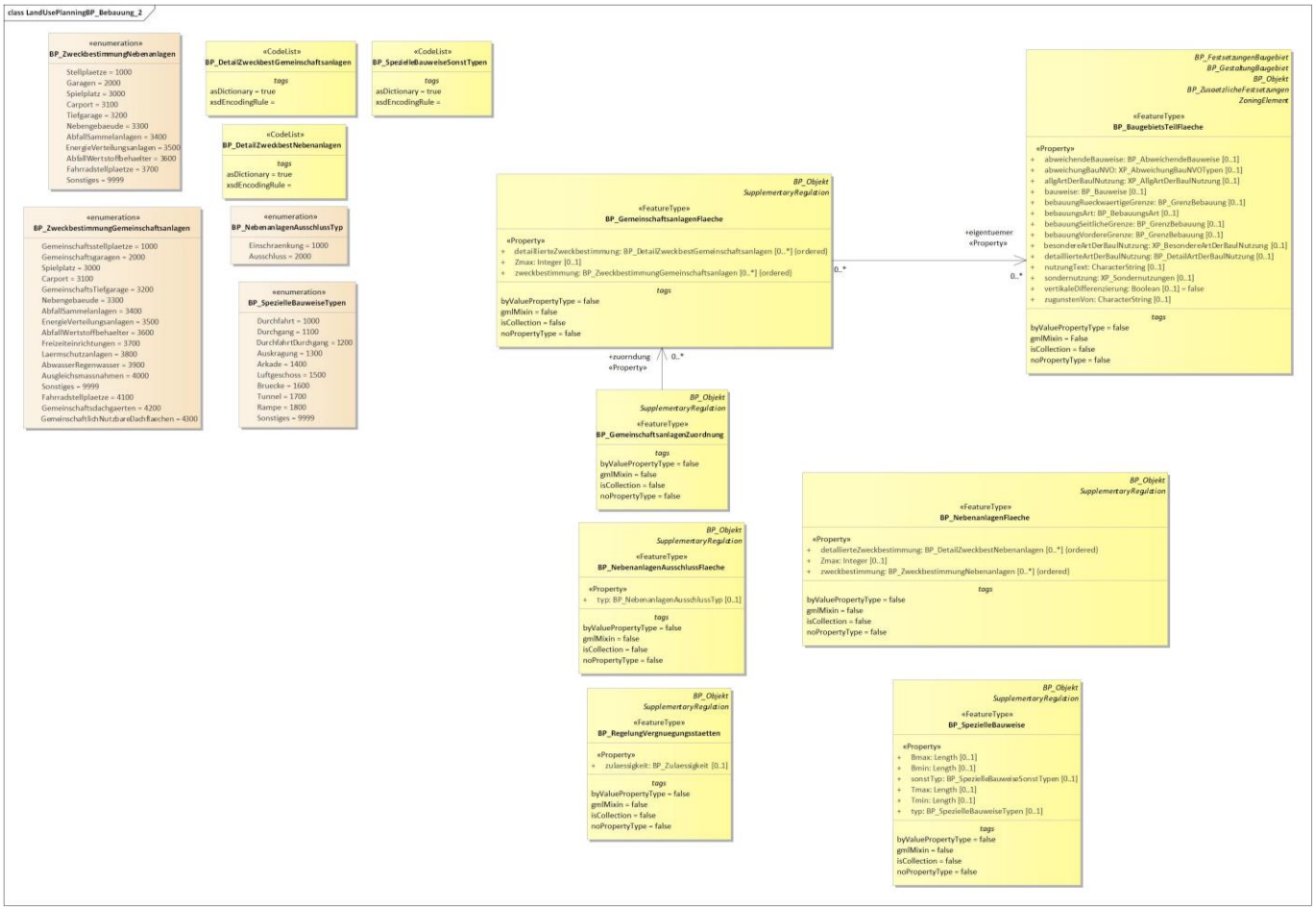


Diagram 6 - LandUsePlanningBP_Bebauung_2

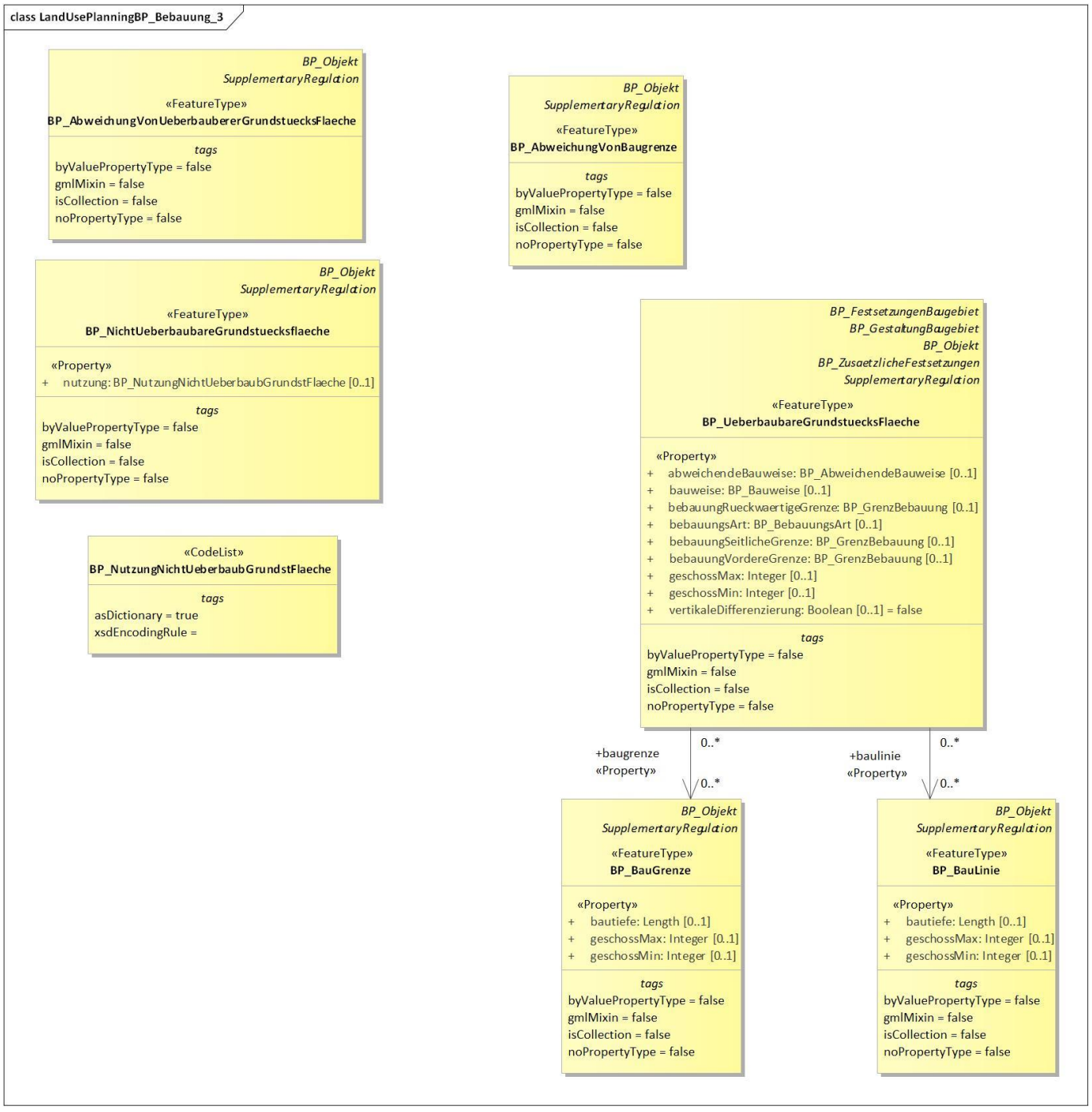


Diagram 7 - LandUsePlanningBP_Bebauung_3

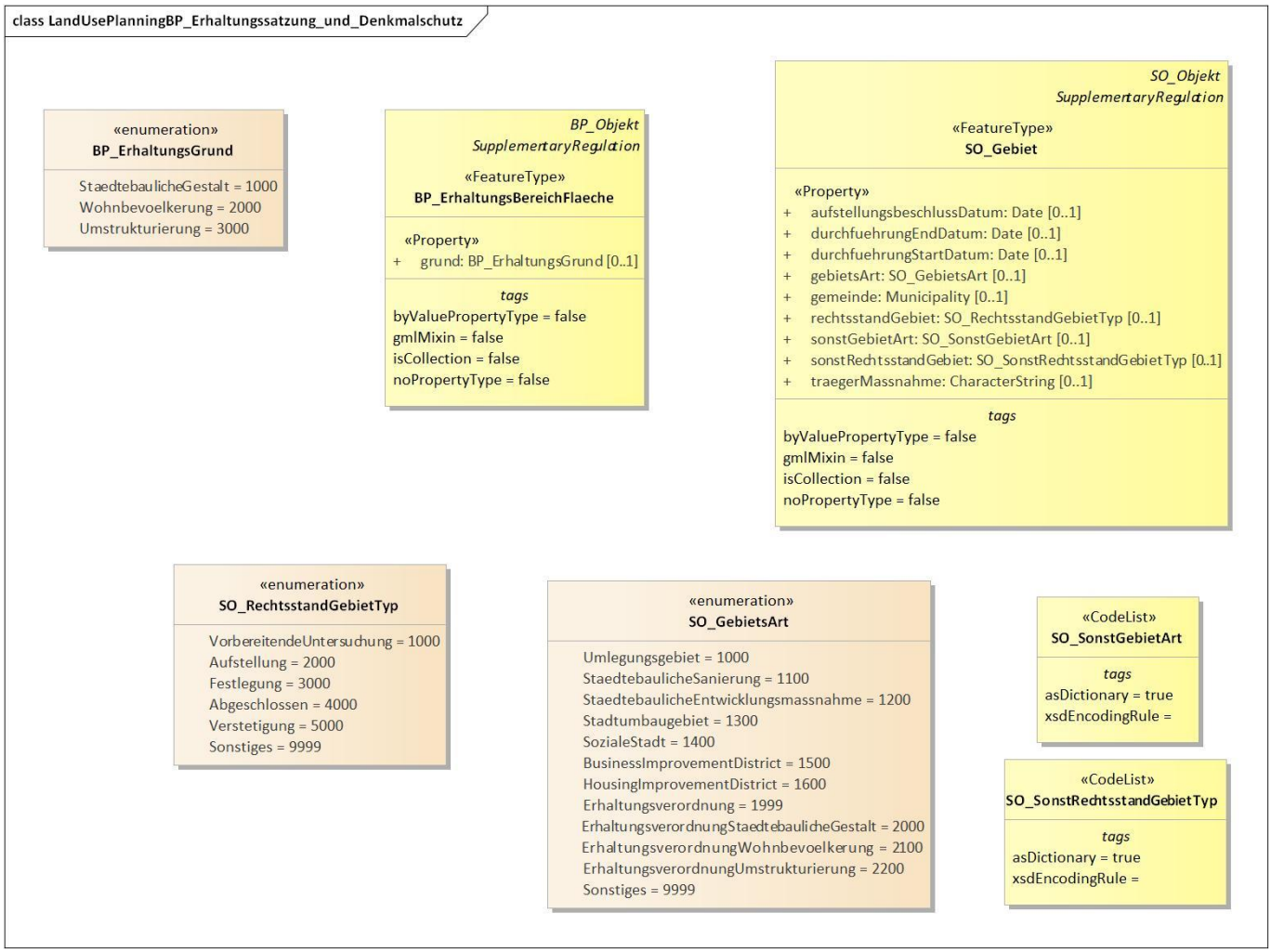


Diagram 8 - LandUsePlanningBP_Erhaltungssatzung_und_Denkmalschutz

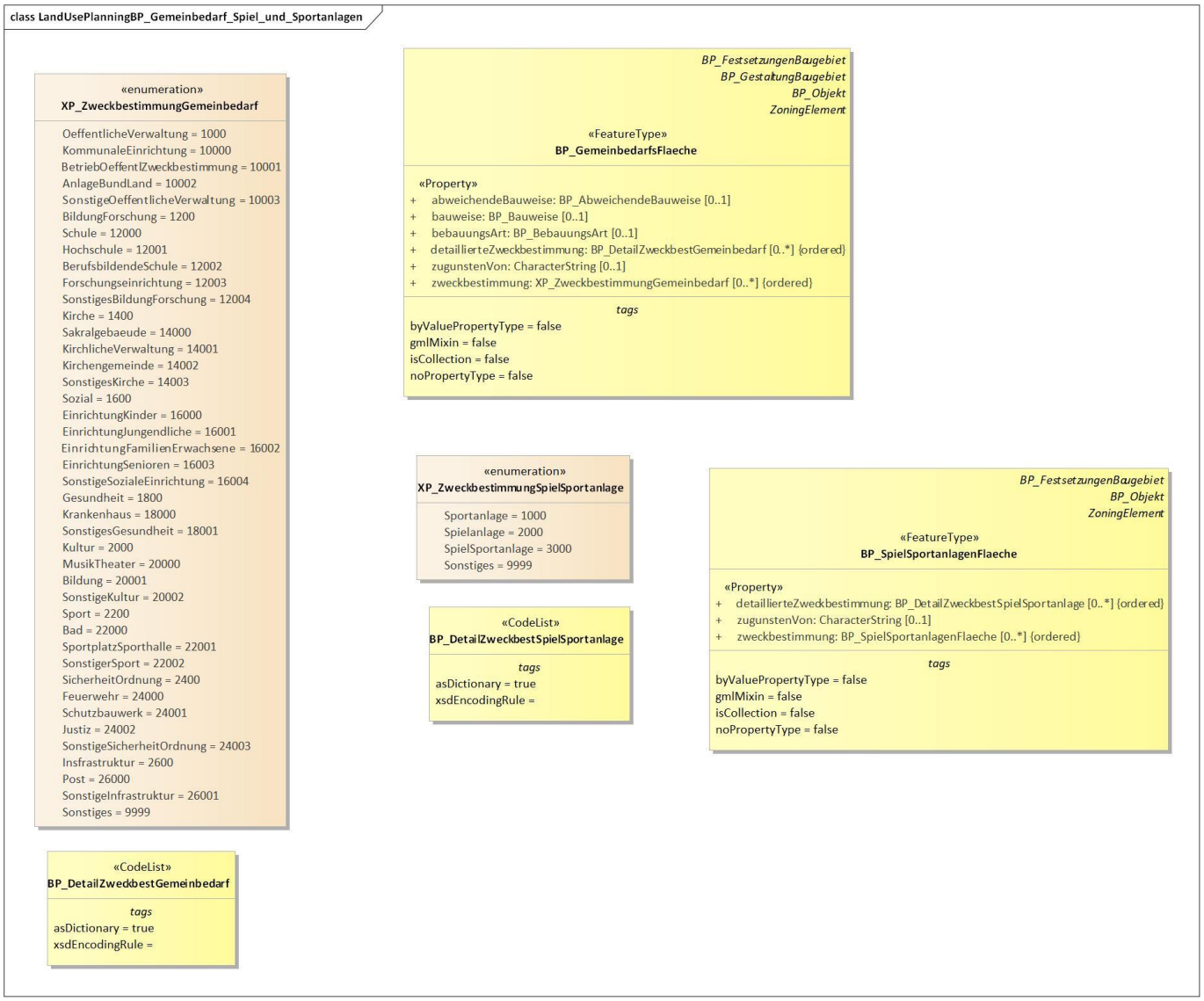


Diagram 9 - LandUsePlanningBP_Gemeinbedarf_Spiel_und_Sportanlagen

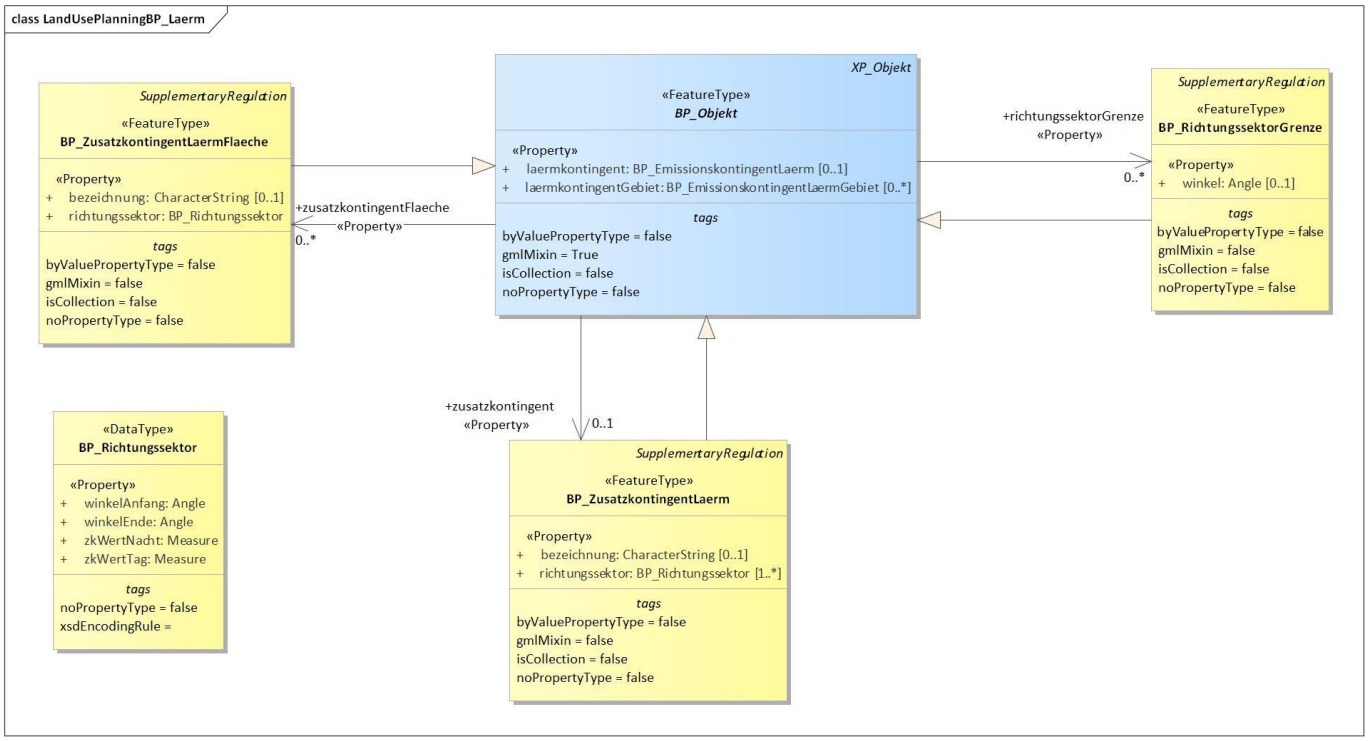


Diagram 10 - LandUsePlanningBP_Laerm

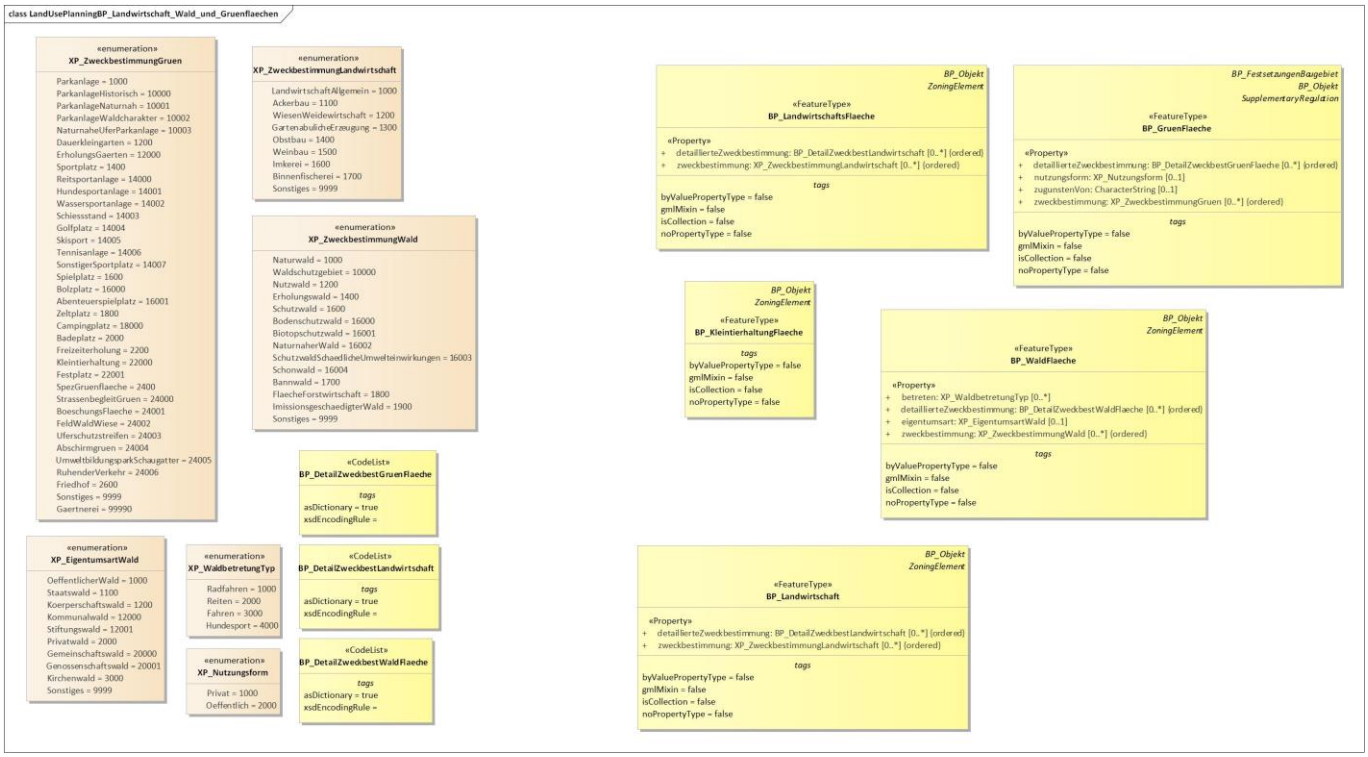


Diagram 11 - LandUsePlanningBP_Landwirtschaft_Wald_und_Gruenflaechen

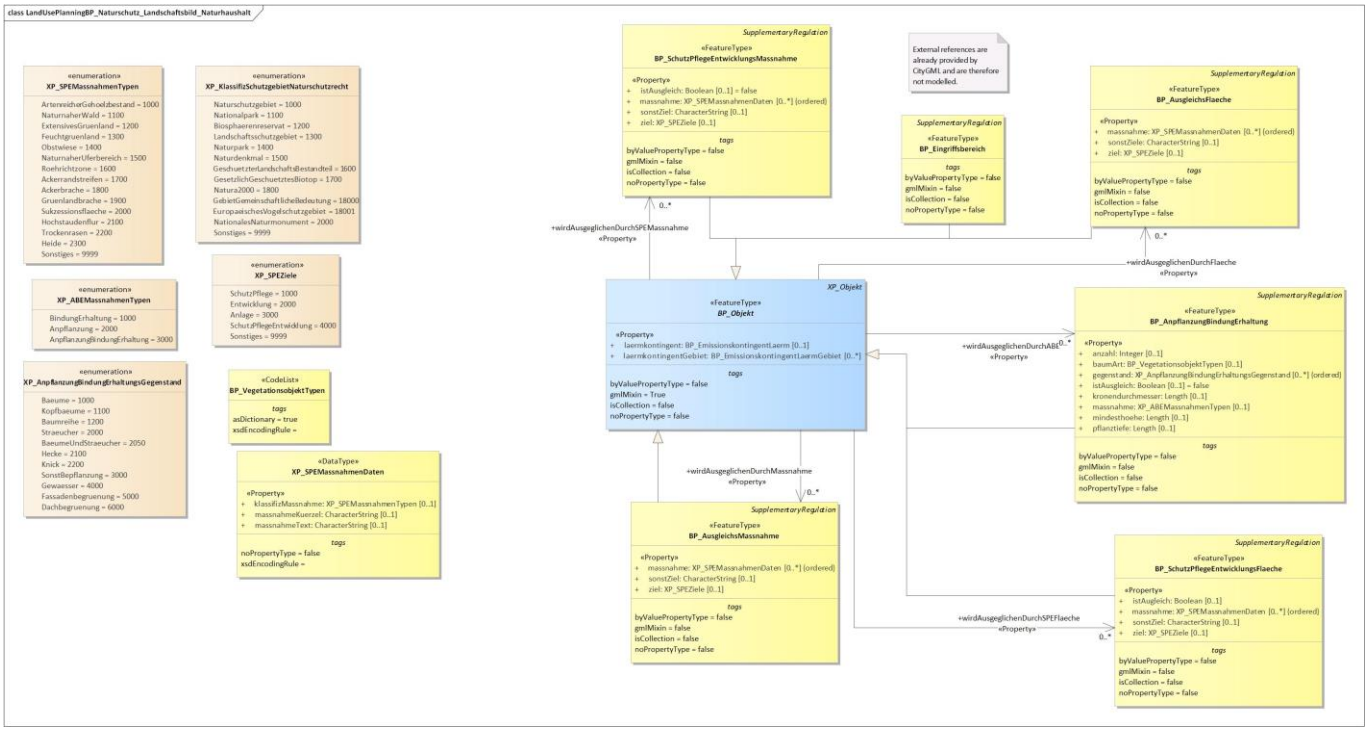


Diagram 12 - LandUsePlanningBP_Naturschutz_Landschaftsbild_Naturhaushalt

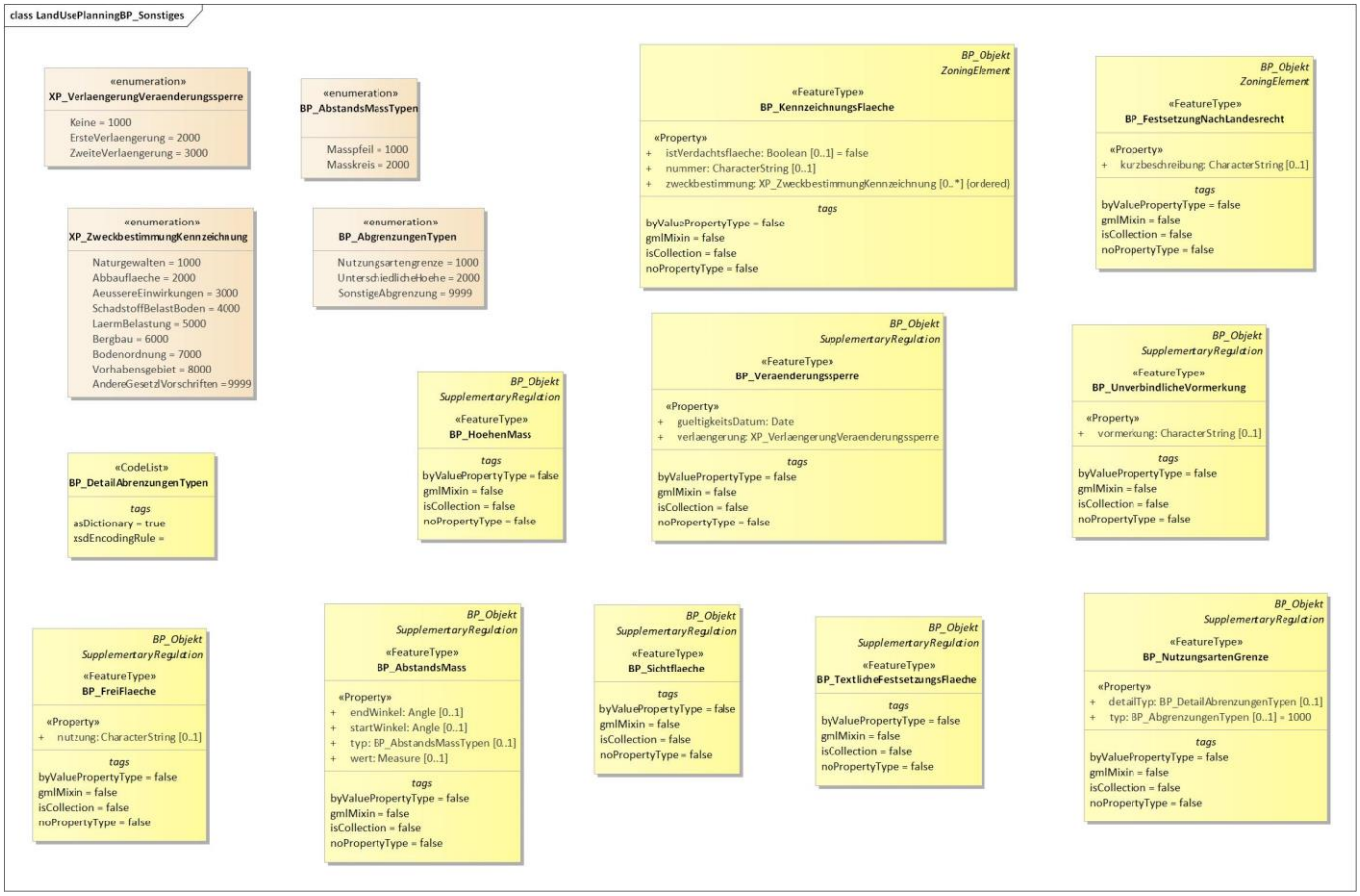


Diagram 13 - LandUsePlanningBP_Sonstiges

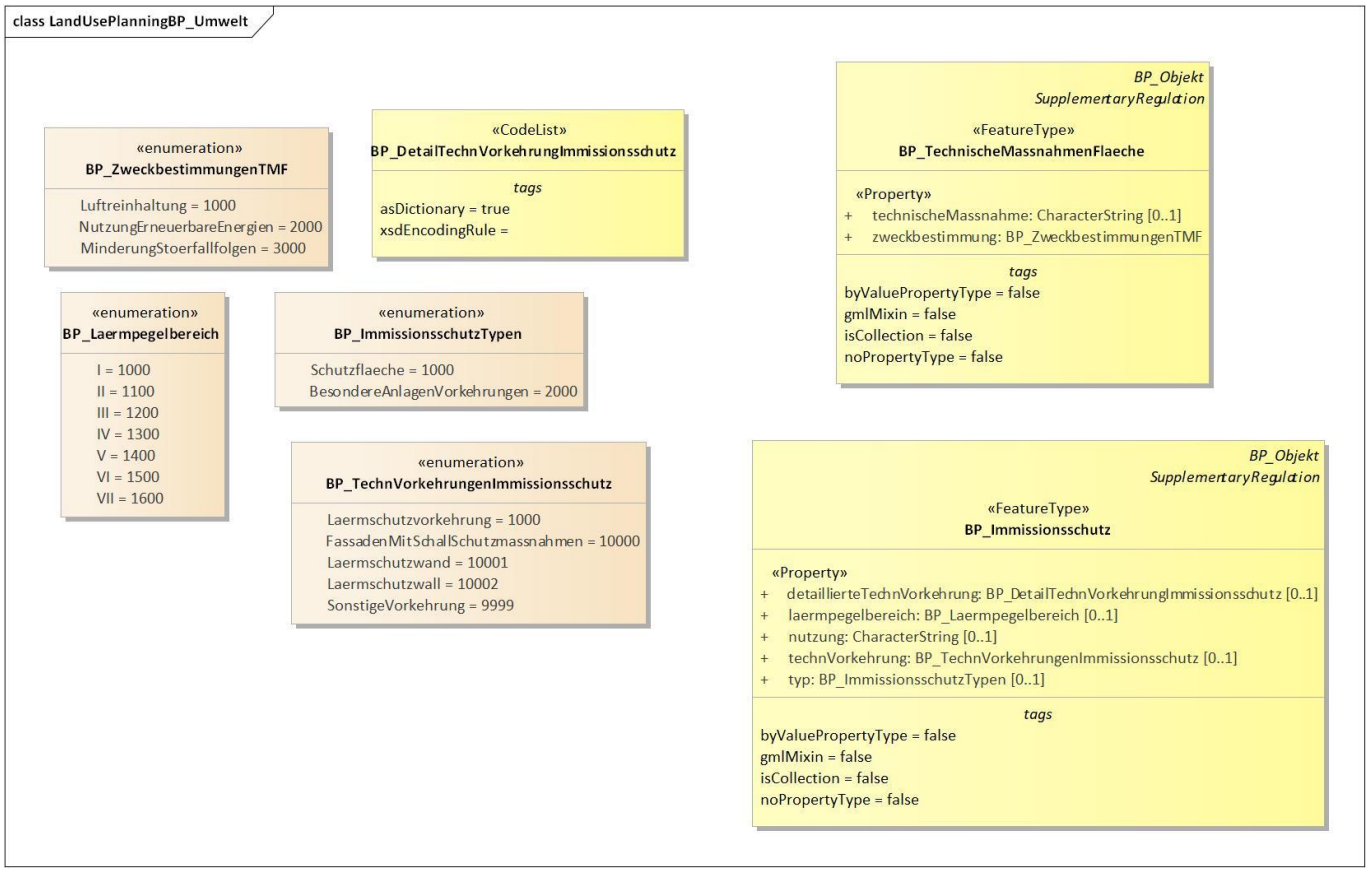


Diagram 14 - LandUsePlanningBP_Umwelt

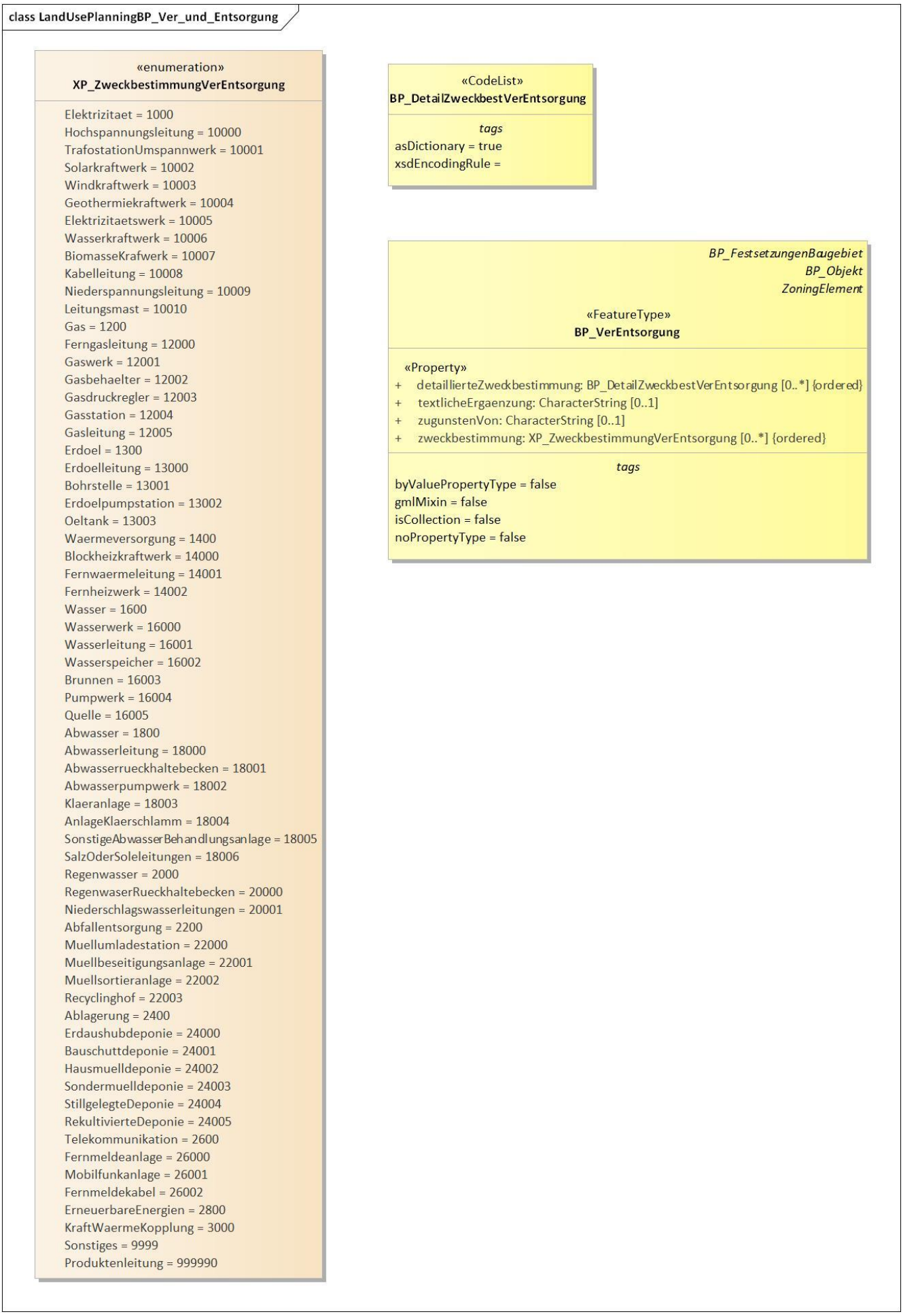

class LandUsePlanningBP_Ver_und_Entsorgung
«enumeration»
XP_ZweckbestimmungVerEntsorgung
Elektrizitaet = 1000
Hochspannungsleitung = 10000
TrafostationUmspannwerk = 10001
Solarkraftwerk = 10002
Windkraftwerk = 10003
Geothermiekraftwerk = 10004
Elektrizitaetswerk = 10005
Wasserkraftwerk = 10006
BiomasseKrafwerk = 10007
Kabelleitung = 10008
Niederspannungsleitung = 10009
Leitungsmast = 10010
Gas = 1200
Ferngasleitung = 12000
Gaswerk = 12001
Gasbehaelter = 12002
Gasdruckregler = 12003
Gasstation = 12004
Gasleitung = 12005
Erdoel = 1300
Erdoelleitung = 13000
Bohrstelle = 13001
Erdoelpumpstation = 13002
Oeltank = 13003
Waermeversorgung = 1400
Blockheizkraftwerk = 14000
Fernwaermeleitung = 14001
Fernheizwerk = 14002
Wasser = 1600
Wasserwerk = 16000
Wasserleitung = 16001
Wasserspeicher = 16002
Brunnen = 16003
Pumpwerk = 16004
Quelle = 16005
Abwasser = 1800
Abwasserleitung = 18000
Abwasserrueckhaltebecken = 18001
Abwasserpumpwerk = 18002
Klaeranlage = 18003
AnlageKlaerschlamm = 18004
SonstigeAbwasserBehandlungsanlage = 18005
SalzOderSoleleitungen = 18006
Regenwasser = 2000
RegenwaserRueckhaltebecken = 20000
Niederschlagswasserleitungen = 20001
Abfallentsorgung = 2200
Muellumladestation = 22000
Muellbeseitigungsanlage = 22001
Muellsortieranlage = 22002
Recyclinghof = 22003
Ablagerung = 2400
Erdaushubdeponie = 24000
Bauschuttdeponie = 24001
Hausmuelldeponie = 24002
Sondermuelldeponie = 24003
StillgelegteDeponie = 24004
RekultivierteDeponie = 24005
Telekommunikation = 2600
Fernmeldeanlage = 26000
Mobilfunkanlage = 26001
Fernmeldekabel = 26002
ErneuerbareEnergien = 2800
KraftWaermeKopplung = 3000
Sonstiges = 9999
Produktenleitung = 999990
«CodeList»
BP_DetailZweckbestVerEntsorgung
tags
asDictionary = true
xsdEncodingRule =
BP_FestsetzungenBaugebiet
BP_Objekt
ZoningElement
«FeatureType»
BP_VerEntsorgung
«Property»
+ detaillierteZweckbestimmung: BP_DetailZweckbestVerEntsorgung [0..*] {ordered}
+ textlicheErgaenzung: CharacterString [0..1]
+ zugunstenVon: CharacterString [0..1]
+ zweckbestimmung: XP_ZweckbestimmungVerEntsorgung [0..*] {ordered}
tags
byValuePropertyType = false
gmlMixin = false
isCollection = false
noPropertyType = false

Diagram 15 - LandUsePlanningBP_Ver_und_Entsorgung

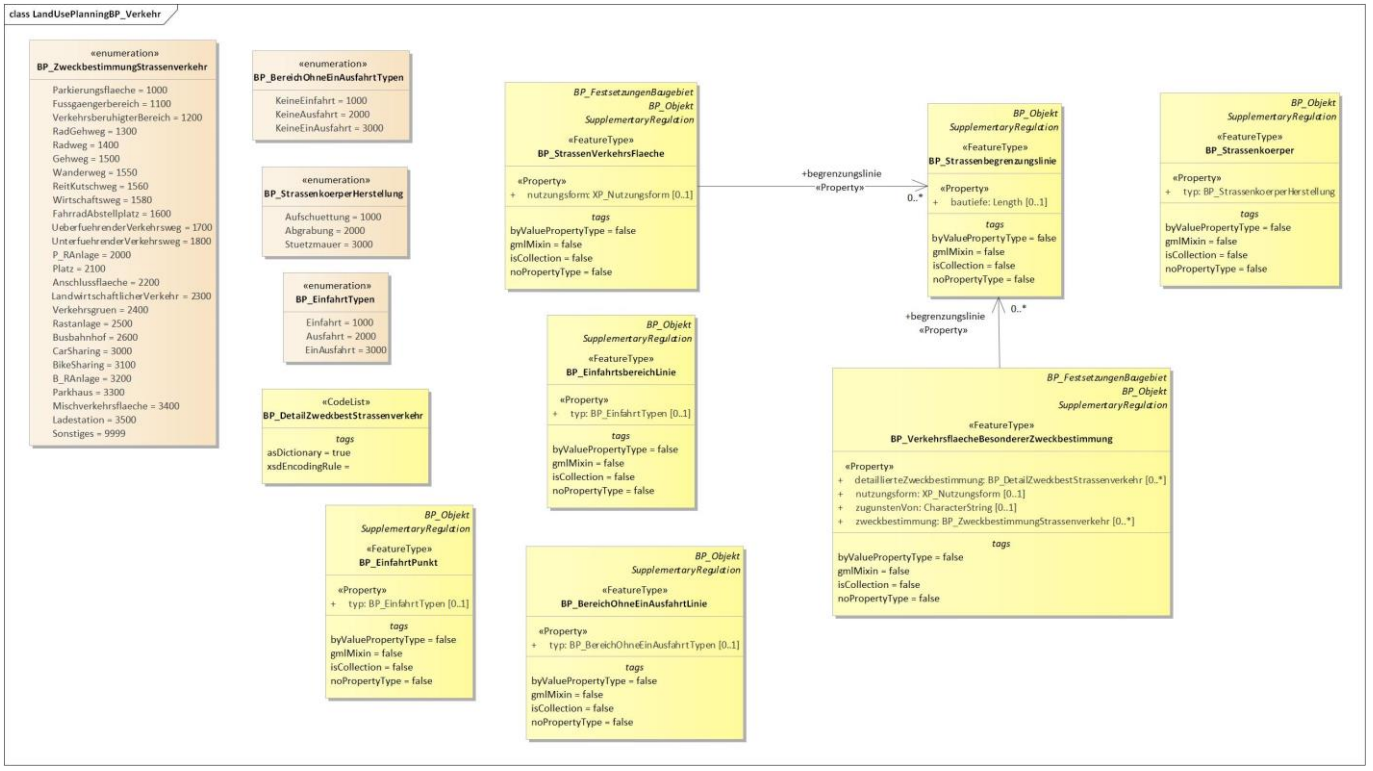


Diagram 16 - LandUsePlanningBP_Verkehr

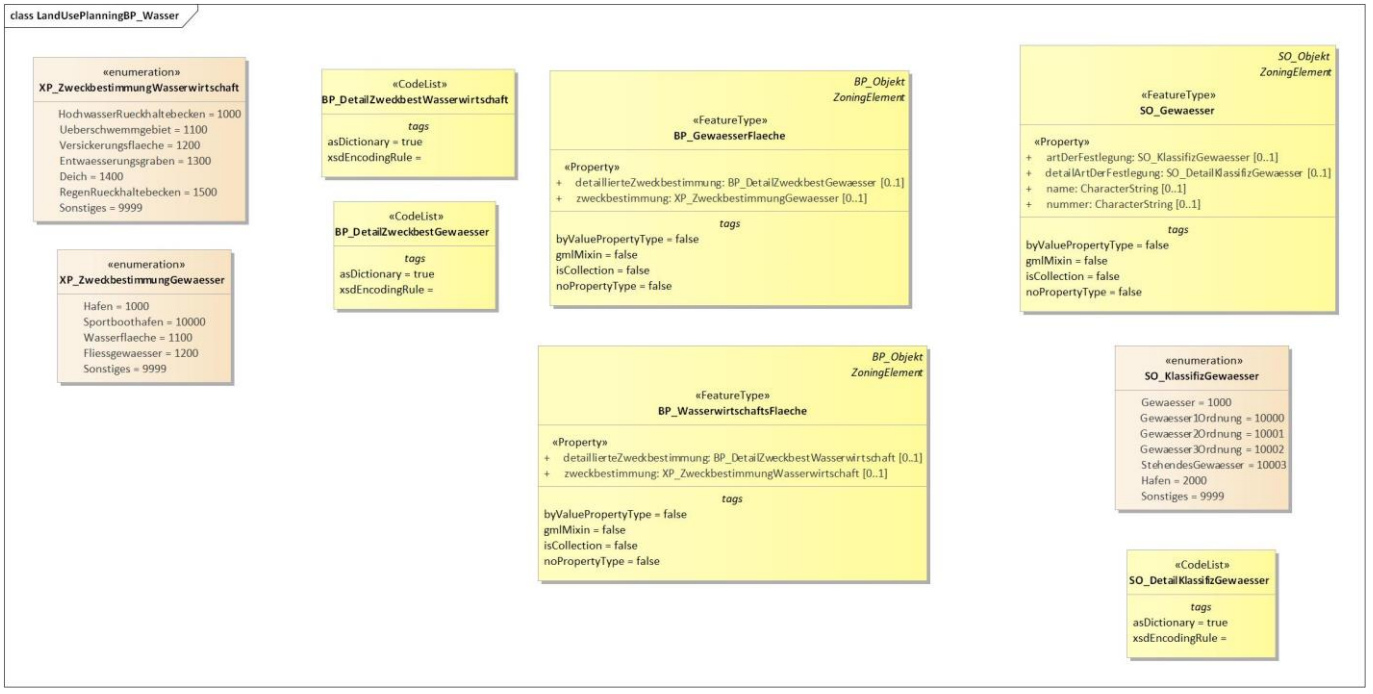


Diagram 17 - LandUsePlanningBP_Wasser

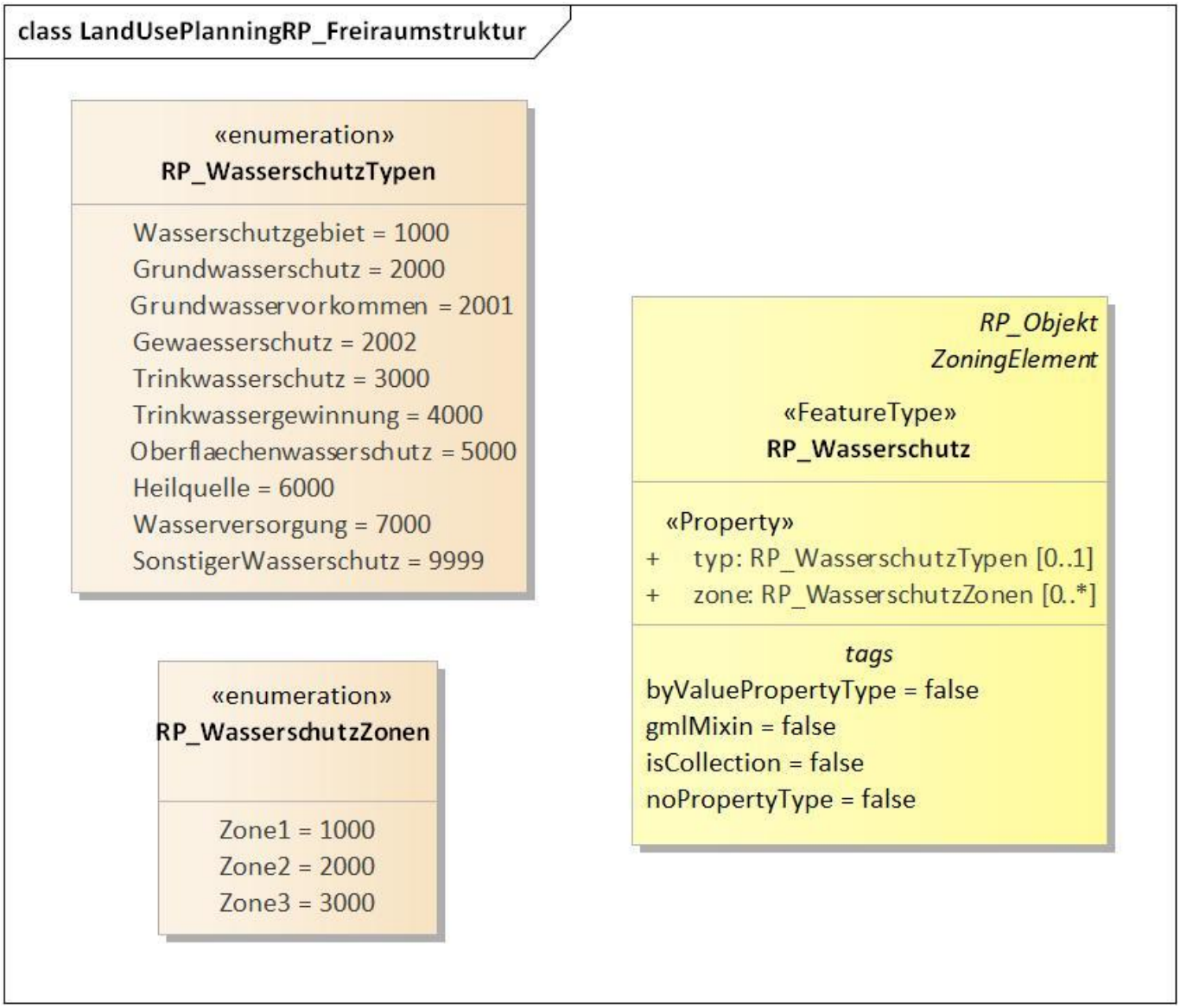


Diagram 18 - LandUsePlanningRP_Freiraumstruktur

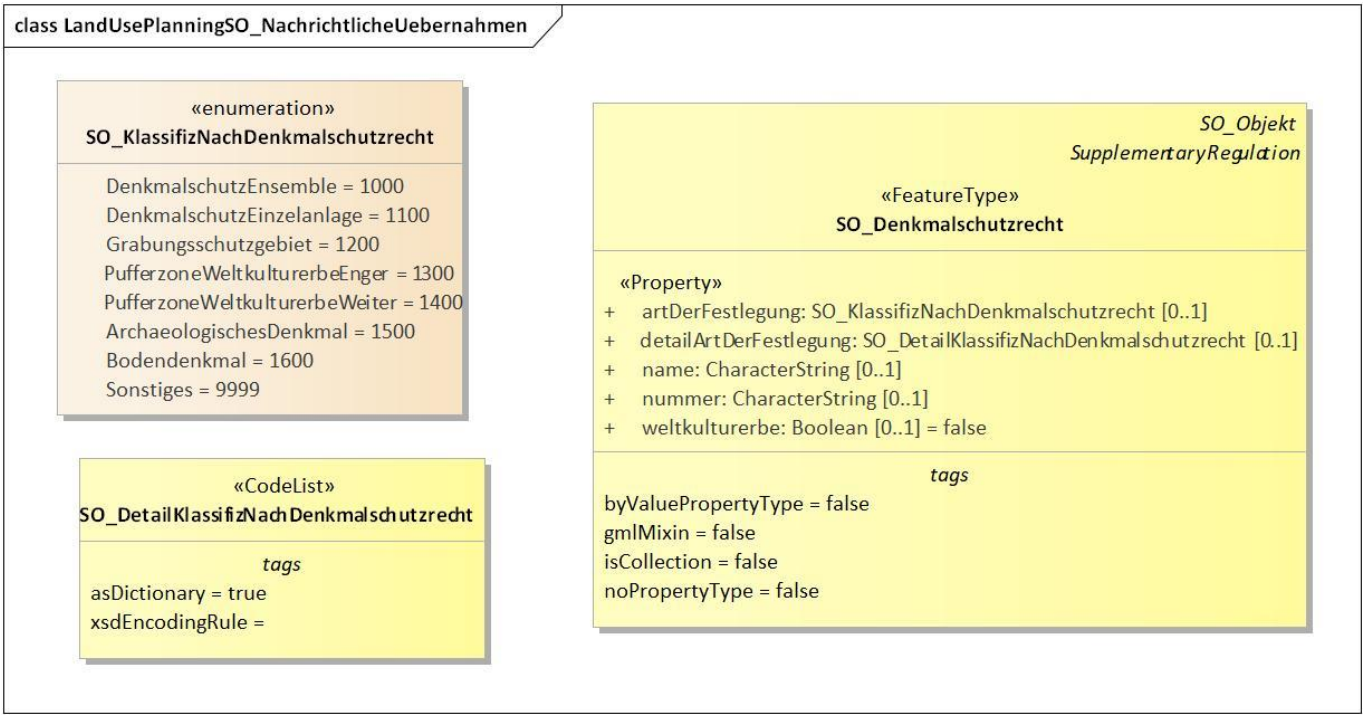


Diagram 19 - LandUsePlanningSO_NachrichtlicheUebernahmen

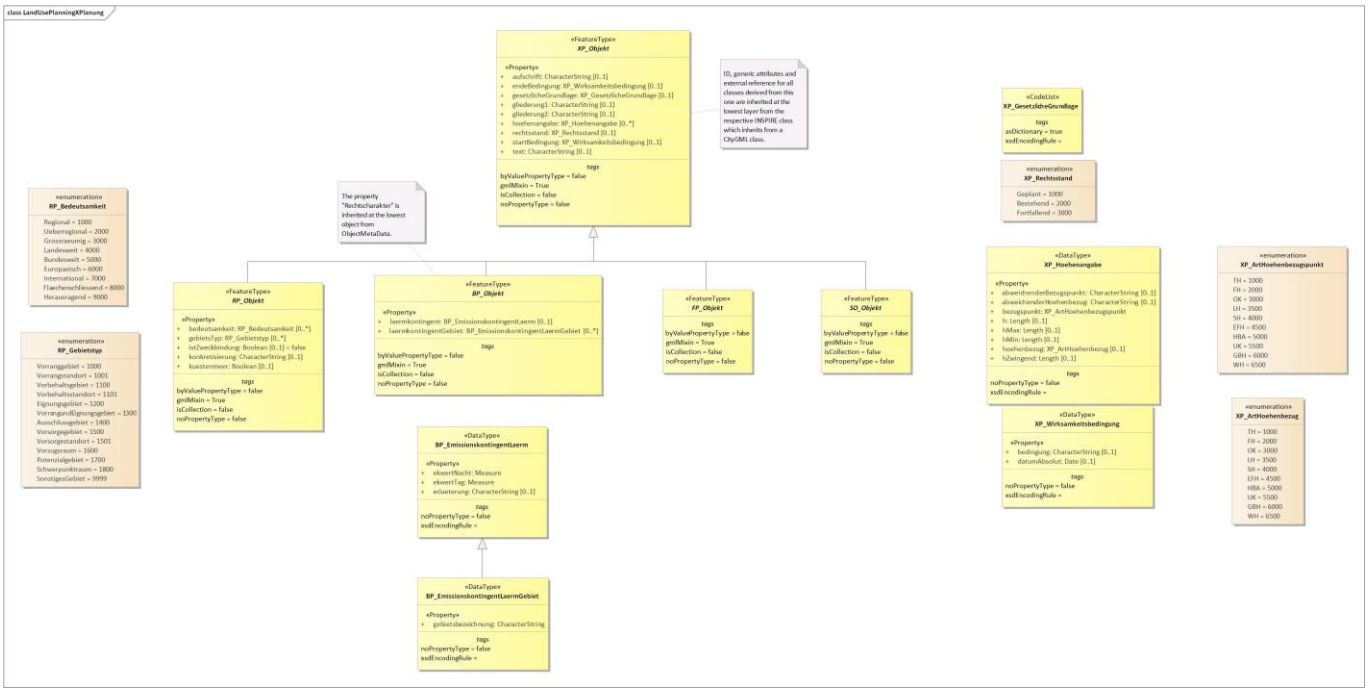


Diagram 20 - LandUsePlanningXPlanung

1.5.1 BP_AbgrabungsFlaeche

BP_AbgrabungsFlaeche

Definition:	From the XPlanung 5.2 specification: Flächen für Aufschüttungen, Abgrabungen oder für die Gewinnung von Bodenschätzen (§9, Abs. 1, Nr. 17 BauGB)). Hier: Flächen für Abgrabungen und die Gewinnung von Bodenschätzen.

Subtype of:	ZoningElement
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	abbaugut
Definition:	From the XPlanung 5.2 specification: Bezeichnung des Abbauguts.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.2 BP_AbstandsFlaeche

BP_AbstandsFlaeche	
Definition:	From the XPlanung 5.2 specification: Festsetzung eines vom Bauordnungsrecht abweichenden Maßes der Tiefe der Abstandsfläche gemäß § 9 Abs 1. Nr. 2a BauGB.
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	tiefe
Voidable:	false
Multiplicity:	0..1
Value type:	Length

1.5.3 BP_AbstandsMass

BP_AbstandsMass	
Definition:	From the XPlanung 5.2 specification: Darstellung von Maßpfeilen oder Maßkreisen in BPlänen, um eine eindeutige Vermassung einzelner Festsetzungen zu erreichen. Bei Masspfeilen (typ == 1000) sollte das Geometrie-Attribut position nur eine einfache Linien (gml:LineString mit 2 Punkten) enthalten

	Bei Maßkreisen (typ == 2000) sollte position nur einen einfachen Kreisbogen (gml:Curve mit genau einem gml:Arc enthalten. In der nächsten Hauptversion von XPlanGML werden diese Empfehlungen zu verpflichtenden Konformitätsbedingungen.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	endWinkel
Definition:	From the XPlanung 5.2 specification: Endwinkel für die Planarstellung des Abstandsmaßes (nur relevant für Maßkreise). Die Winkelwerte beziehen sich auf den Rechtswert (Ost-Richtung).
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	startWinkel
Definition:	From the XPlanung 5.2 specification: Startwinkel für die Plandarstellung des Abstandsmaßes (nur relevant für Maßkreise). Die Winkelwerte beziehen sich auf den Rechtswert (Ost-Richtung).
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ der Massangabe (Maßpfeil oder Maßkreis).
Voidable:	false
Multiplicity:	0..1
Value type:	BP_AbstandsMassTypen (enumeration)
Values	1000 **Masspfeil** 2000 **Masskreis**

Attribute:

Name:	wert
Definition:	From the XPlanung 5.2 specification: Wertangabe des Abstandsmaßes. Bei Maßpfeilen (typ == 1000) enthält das Attribut die Länge des Maßpfeilen (uom = "m"), bei Maßkreisen den von startWinkel und endWinkel eingeschlossenen Winkel (uom = "grad").
Voidable:	false
Multiplicity:	0..1
Value type:	Measure

1.5.4 BP_AbweichungVonBaugrenze

BP_AbweichungVonBaugrenze

Definition:	From the XPlanung 5.2 specification: Linienhafte Festlegung des Umfangs der Abweichung von der Baugrenze (§23 Abs. 3 Satz 3 BauNVO).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.5 BP_AbweichungVonUeberbaubererGrundstuecksFlaeche

BP_AbweichungVonUeberbaubererGrundstuecksFlaeche

Definition:	From the XPlanung 5.2 specification: Flächenhafte Festlegung des Umfangs der Abweichung von der überbaubaren Grundstücksfläche (§23 Abs. 3 Satz 3 BauNVO).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.6 BP_AnpflanzungBindungErhaltung

BP_AnpflanzungBindungErhaltung

Definition:	From the XPlanung 5.2 specification: Festsetzung des Anpflanzens von Bäumen, Sträuchern und sonstigen Bepflanzungen; Festsetzung von Bindungen für Bepflanzungen und für die Erhaltung von Bäumen, Sträuchern und sonstigen

	Bepflanzungen sowie von Gewässern; (§9 Abs. 1 Nr. 25 und Abs. 4 BauGB).
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	anzahl
Definition:	From the XPlanung 5.2 specification:
	Anzahl der anzupflanzenden Objekte.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	baumArt
Definition:	From the XPlanung 5.2 specification:
	Textliche Spezifikation einer Baumart.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_VegetationsobjektTypen (code list)
Attribute:	
Name:	gegenstand
Definition:	From the XPlanung 5.2 specification:
	Gegenstand der Maßnahme.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_AnpflanzungBindungErhaltungsGegenstand (enumeration)
Values	1000 **Baeume** 1100 **Kopfbaeume** 1200 **Baumreihe** 2000 **Straeucher** 2050 **BaeumeUndStraeucher** 2100 **Hecke**

2200	**Knick**
3000	**SonstBepflanzung**
4000	**Gewaesser**
5000	**Fassadenbegruenung**
6000	**Dachbegruenung**

Attribute:

Name: istAusgleich

Definition: From the XPlanung 5.2 specification:

Gibt an, ob die Fläche oder Maßnahme zum Ausgleich von Eingriffen genutzt wird.

Voidable: false

Multiplicity: 0..1

Initial value: false

Value type: Boolean

Attribute:

Name: kronendurchmesser

Definition: From the XPlanung 5.2 specification:

Durchmesser der Baumkrone bei zu erhaltenden Bäumen.

Voidable: false

Multiplicity: 0..1

Value type: Length

Attribute:

Name: massnahme

Definition: From the XPlanung 5.2 specification:

Art der Maßnahme.

Voidable: false

Multiplicity: 0..1

Value type: XP_ABEMassnahmenTypen (enumeration)

Values

1000	**BindungErhaltung**
2000	**Anpflanzung**
3000	**AnpflanzungBindungErhaltung**

Attribute:

Name:	mindesthoehe
Definition:	From the XPlanung 5.2 specification: Mindesthöhe des Gegenstands der Festsetzung.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	pflanztiefe
Definition:	From the XPlanung 5.2 specification: Pflanztiefe.
Voidable:	false
Multiplicity:	0..1
Value type:	Length

1.5.7 BP_AufschuettungsFlaeche

BP_AufschuettungsFlaeche	
Definition:	From the XPlanung 5.2 specification: Flächen für Aufschüttungen, Abgrabungen oder für die Gewinnung von Bodenschätzen (§ 9 Abs. 1 Nr. 17 und Abs. 6 BauGB). Hier: Flächen für Aufschüttungen
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type
Attribute:	
Name:	aufschuettungsmaterial
Definition:	From the XPlanung 5.2 specification: Bezeichnung des aufgeschütteten Materials.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.8 BP_AusgleichsFlaeche

BP_AusgleichsFlaeche	
Definition:	From the XPlanung 5.2 specification:

	Festsetzung einer Fläche zum Ausgleich im Sinne des § 1a Abs.3 und §9 Abs. 1a BauGB.
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	massnahme
Definition:	From the XPlanung 5.2 specification:
	Auf der Fläche durchzuführende Maßnahmen.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_SPEMassnahmenDaten (data type)
Attribute:	
Name:	sonstZiele
Definition:	From the XPlanung 5.2 specification:
	Textlich formuliertes Ziel, wenn das Attribut ziel den Wert 9999 (Sonstiges) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	ziel
Definition:	From the XPlanung 5.2 specification:
	Ziel der Ausgleichsmaßnahme.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_SPEZiele (enumeration)
Values	1000 **SchutzPflege** 2000 **Entwicklung** 3000 **Anlage** 4000 **SchutzPflegeEntwicklung** 9999 **Sonstiges**

1.5.9 BP_AusgleichsMassnahme

BP_AusgleichsMassnahme

Definition:	From the XPlanung 5.2 specification: Festsetzung einer Einzelmaßnahme zum Ausgleich im Sinne des § 1a Abs.3 und §9 Abs. 1a BauGB.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	massnahme
Definition:	From the XPlanung 5.2 specification: Durchzuführende Ausgleichsmaßnahme.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_SPEMassnahmenDaten (data type)

Attribute:

Name:	sonstZiel
Definition:	From the XPlanung 5.2 specification: Textlich formuliertes Ziel, wenn das Attribut ziel den Wert 9999 (Sonstiges) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	ziel
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	0..1
Value type:	XP_SPEZiele (enumeration)

Values

1000	**SchutzPflege**
2000	**Entwicklung**
3000	**Anlage**
4000	**SchutzPflegeEntwicklung**

	9999	**Sonstiges**	

1.5.10 BP_BauGrenze

BP_BauGrenze	
Definition:	From the XPlanung 5.2 specification: Festsetzung einer Baugrenze (§9 Abs. 1 Nr. 2 BauGB, §22 und 23 BauNVO). Über die Attribute geschossMin und geschossMax kann die Festsetzung auf einen Bereich von Geschossen beschränkt werden. Wenn eine Einschränkung der Festsetzung durch expliziter Höhenangaben erfolgen soll, ist dazu die Oberklassen-Relation hoehenangabe auf den komplexen Datentyp XP_Hoehenangabe zu verwenden. Durch die Digitalisierungsreihenfolge der Linienstützpunkte muss sichergestellt sein, dass die überbaute Fläche (BP_UeberbaubareGrundstuecksFlaeche) relativ zur Laufrichtung auf der linken Seite liegt.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	bautiefe
Definition:	From the XPlanung 5.2 specification: Angabe einer Bautiefe.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	geschossMax
Definition:	From the XPlanung 5.2 specification: Gibt bei geschossweiser Festsetzung die Nummer des Geschosses an, bis zu der die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse ab einschl. "geschossMin".
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	

Name:	geschossMin
Definition:	From the XPlanung 5.2 specification: Gibt bei geschossweiser Festsetzung die Nummer des Geschosses an, ab den die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse bis einschl. "geschossMax".
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.5.11 BP_BauLinie

BP_BauLinie	
Definition:	From the XPlanung 5.2 specification: Festsetzung einer Baulinie (§9 Abs. 1 Nr. 2 BauGB, §22 und 23 BauNVO). Über die Attribute geschossMin und geschossMax kann die Festsetzung auf einen Bereich von Geschossen beschränkt werden. Wenn eine Einschränkung der Festsetzung durch explizite Höhenangaben erfolgen soll, ist dazu die Oberklassen-Relation hoehenangabe auf den komplexen Datentyp XP_Hoehenangabe zu verwenden. Durch die Digitalisierungsreihenfolge der Linienstützpunkte muss sichergestellt sein, dass die überbaute Fläche (BP_UeberbaubareGrundstuecksFlaeche) relativ zur Laufrichtung auf der linken Seite liegt.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	bautiefe
Definition:	From the XPlanung 5.2 specification: Angabe einer Bautiefe.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	geschossMax
Definition:	From the XPlanung 5.2 specification:

	Gibt bei geschossweiser Feststzung die Nummer des Geschosses an, bis zu der die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse ab einschl. "geschossMin".
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	geschossMin
Definition:	From the XPlanung 5.2 specification: Gibt bei geschossweiser Festsetzung die Nummer des Geschosses an, ab den die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse bis einschl. "geschossMax".
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.5.12 BP_BaugebietsTeilFlaeche

BP_BaugebietsTeilFlaeche	
Definition:	From the XPlanung 5.2 specification: Teil eines Baugebiets mit einheitlicher Art der baulichen Nutzung. Das Maß der baulichen Nutzung sowie Festsetzungen zur Bauweise oder Grenzbebauung können innerhalb einer BP_BaugebietsTeilFlaeche unterschiedlich sein (BP_UeberbaubareGrundstueckeFlaeche).
Subtype of:	ZoningElement BP_Objekt BP_GestaltungBaugebiet BP_ZusaetzlicheFestsetzungen BP_FestsetzungenBaugebiet
Type:	Feature type
Attribute:	
Name:	abweichendeBauweise
Definition:	From the XPlanung 5.2 specification: Nähere Bezeichnung einer "Abweichenden Bauweise" ("bauweise" == 3000).

Voidable: false

Multiplicity: 0..1

Value type: BP_AbweichendeBauweise (code list)

Attribute:

Name: abweichungBauNVO

Definition: From the XPlanung 5.2 specification:

Art der zulässigen Abweichung von der BauNVO.

Voidable: false

Multiplicity: 0..1

Value type: XP_AbweichungBauNVOTypen (enumeration)

Values

1000	**EinschraenkungNutzung**
2000	**AusschlussNutzung**
3000	**AusweitungNutzung**
9999	**SonstAbweichung**

Attribute:

Name: allgArtDerBaulNutzung

Definition: From the XPlanung 5.2 specification:

Spezifikation der allgemeinen Art der baulichen Nutzung.

Voidable: false

Multiplicity: 0..1

Value type: XP_AllgArtDerBaulNutzung (enumeration)

Values

1000	**WohnBauflaeche**
2000	**GemischteBauflaeche**
3000	**GewerblicheBauflaeche**
4000	**SonderBauflaeche**
9999	**SonstigeBauflaeche**

Attribute:

Name: bauweise

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_Bauweise (enumeration)

Values

1000	**OffeneBauweise**
2000	**GeschlosseneBauweise**
3000	**AbweichendeBauweise**

Attribute:

Name: bebauungRueckwaertigeGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der rückwärtigen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungSeitlicheGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der seitlichen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungVordereGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der vorderen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungsArt

Definition: From the XPlanung 5.2 specification:

Detaillierte Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_BebauungsArt (enumeration)

Values

1000	**Einzelhaeuser**
2000	**Doppelhaeuser**
3000	**Hausgruppen**
4000	**EinzelDoppelhaeuser**
5000	**EinzelhaeuserHausgruppen**
6000	**DoppelhaeuserHausgruppen**
7000	**Reihenhaeuser**
8000	**EinzelhaeuserDoppelhaeuserHausgruppen**

Attribute:

Name: besondereArtDerBaulNutzung

Definition: From the XPlanung 5.2 specification:

Festsetzung der Art der baulichen Nutzung (§9, Abs. 1, Nr. 1 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: XP_BesondereArtDerBaulNutzung (enumeration)

Values

1000	**Kleinsiedlungsgebiet**
1100	**ReinesWohngebiet**
1200	**AllgWohngebiet**
1300	**BesonderesWohngebiet**
1400	**Dorfgebiet**

1500	**Mischgebiet**
1550	**UrbanesGebiet**
1600	**Kerngebiet**
1700	**Gewerbegebiet**
1800	**Industriegebiet**
2000	**SondergebietErholung**
2100	**SondergebietSonst**
3000	**Wochenendhausgebiet**
4000	**Sondergebiet**
9999	**SonstigesGebiet**

Attribute:

Name: detaillierteArtDerBaulNutzung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere Nutzungsart.

Voidable: false

Multiplicity: 0..1

Value type: BP_DetailArtDerBaulNutzung (code list)

Attribute:

Name: nutzungText

Definition: From the XPlanung 5.2 specification:

Bei Nutzungsform "Sondergebiet" ("besondereArtDerBaulNutzung" == 4000): Kurzform der besonderen Art der baulichen Nutzung.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: sondernutzung

Definition: From the XPlanung 5.2 specification:

Differenziert Sondernutzungen nach §10 und §11 der BauNVO von 1977 und 1990. Das Attribut wird nur benutzt, wenn besondereArtDerBaulNutzung unbelegt ist oder einen der Werte 2000 bzw. 2100 hat.

Voidable: false

Multiplicity: 0..1

Value type: XP_Sondernutzungen (enumeration)

Values

1000	**Wochenendhausgebiet**
1100	**Ferienhausgebiet**
1200	**Campingplatzgebiet**
1300	**Kurgebiet**
1400	**SonstSondergebietErholung**
1500	**Einzelhandelsgebiet**
1600	**GrossflaechigerEinzelhandel**
16000	**Ladengebiet**
16001	**Einkaufszentrum**
16002	**SonstGrossflEinzelhandel**
1700	**Verkehrsuebungsplatz**
1800	**Hafengebiet**
1900	**SondergebietErneuerbareEnergie**
2000	**SondergebietMilitaer**
2100	**SondergebietLandwirtschaft**
2200	**SondergebietSport**
2300	**SondergebietGesundheitSoziales**
23000	**Klinikgebiet**
2400	**Golfplatz**
2500	**SondergebietKultur**
2600	**SondergebietTourismus**
2700	**SondergebietBueroUndVerwaltung**
2720	**SondergebietJustiz**
2800	**SondergebietHochschuleForschung**
2900	**SondergebietMesse**
9999	**SondergebietAndereNutzungen**

Attribute:

Name: vertikaleDifferenzierung

Definition: From the XPlanung 5.2 specification:

	Gibt an, ob eine vertikale Differenzierung der Gebäude vorgeschrieben ist.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean
Attribute:	
Name:	zugunstenVon
Definition:	From the XPlanung 5.2 specification: Angabe des Begünstigten einer Ausweisung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.13 BP_BereichOhneEinAusfahrtLinie

BP_BereichOhneEinAusfahrtLinie	
Definition:	From the XPlanung 5.2 specification: Bereich ohne Ein- und Ausfahrt (§ 9 Abs. 1 Nr. 11 und Abs. 6 BauGB). Durch die Digitalisierungsreihenfolge der Linienstützpunkte muss sichergestellt sein, dass der angrenzende Bereich ohne Ein- und Ausfahrt relativ zur Laufrichtung auf der linken Seite liegt.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ des Bereiches ohne Ein- und Ausfahrt.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_BereichOhneEinAusfahrtTypen (enumeration)
Values	1000 **KeineEinfahrt**

2000	**KeineAusfahrt**	
3000	**KeineEinAusfahrt**	

1.5.14 BP_BesondererNutzungszweckFlaeche

BP_BesondererNutzungszweckFlaeche

Definition: From the XPlanung 5.2 specification:

Festsetzung einer Fläche mit besonderem Nutzungszweck, der durch besondere städtebauliche Gründe erfordert wird (§9 Abs. 1 Nr. 9 BauGB).

Subtype of: ZoningElement

BP_Objekt

BP_GestaltungBaugebiet

BP_FestsetzungenBaugebiet

Type: Feature type

Attribute:

Name: abweichendeBauweise

Definition: From the XPlanung 5.2 specification:

Nähere Bezeichnung einer "Abweichenden Bauweise" ("bauweise" == 3000).

Voidable: false

Multiplicity: 0..1

Value type: BP_AbweichendeBauweise (code list)

Attribute:

Name: bauweise

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_Bauweise (enumeration)

Values

1000	**OffeneBauweise**
2000	**GeschlosseneBauweise**
3000	**AbweichendeBauweise**

Attribute:

Name: bebauungsart

Definition: From the XPlanung 5.2 specification:

Detaillierte Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_BebauungsArt (enumeration)

Values

1000	**Einzelhaeuser**
2000	**Doppelhaeuser**
3000	**Hausgruppen**
4000	**EinzelDoppelhaeuser**
5000	**EinzelhaeuserHausgruppen**
6000	**DoppelhaeuserHausgruppen**
7000	**Reihenhaeuser**
8000	**EinzelhaeuserDoppelhaeuserHausgruppen**

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Angabe des besonderen Nutzungszwecks.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

1.5.15 BP_BodenschaetzeFlaeche

BP_BodenschaetzeFlaeche

Definition: From the XPlanung 5.2 specification:

Flächen für Aufschüttungen, Abgrabungen oder für die Gewinnung von Bodenschätzen (§ 9 Abs. 1 Nr. 17 und Abs. 6 BauGB). Hier: Flächen für Gewinnung von Bodenschätzen

Die Klasse wird als veraltet gekennzeichnet und wird in XPlanGML V. 6.0 wegfallen. Es sollte stattdessen die Klasse BP_AbgrabungsFlaeche verwendet werden.

Subtype of: ZoningElement

BP_Objekt

Type:	Feature type
Attribute:	
Name:	abbaugut
Definition:	From the XPlanung 5.2 specification: Bezeichnung des Abbauguts.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.16 BP_Dachgestaltung

BP_Dachgestaltung	
Definition:	From the XPlanung 5.2 specification: Zusammenfassung von Parametern zur Festlegung der zulässigen Dachformen.
Type:	Data type
Attribute:	
Name:	DN
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Dachneigung.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	DNZwingend
Definition:	From the XPlanung 5.2 specification: Zwingend vorgeschriebene Dachneigung.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	DNmax
Definition:	From the XPlanung 5.2 specification:

Maximale Dachneigung bei einer Bereichsangabe. Das Attribut DNmin muss ebenfalls belegt sein..

Voidable: false

Multiplicity: 0..1

Value type: Angle

Attribute:

Name: DNmin

Definition: From the XPlanung 5.2 specification:

Minimale Dachneigung bei einer Bereichsangabe. Das Attribut DNmax muss ebenfalls belegt sein.

Voidable: false

Multiplicity: 0..1

Value type: Angle

Attribute:

Name: dachform

Definition: From the XPlanung 5.2 specification:

Erlaubte Dachform.

Voidable: false

Multiplicity: 0..1

Value type: BP_Dachform (enumeration)

Values

1000	**Flachdach**
2100	**Pultdach**
2200	**VersetztesPultdach**
3000	**GeneigtesDach**
3100	**Satteldach**
3200	**Walmdach**
3300	**Krueppelwalmdach**
3400	**Mansarddach**
3500	**Zeltdach**
3600	**Kegeldach**
3700	**Kuppeldach**
3800	**Sheddach**
3900	**Bogendach**

	4000	**Turmdach**
	4100	**Tonnendach**
	5000	**Mischform**
	9999	**Sonstiges**

Attribute:

Name:	detaillierteDachform
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definiertere detailliertere Dachform.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_DetailDachform (code list)

1.5.17 BP_EinfahrtPunkt

BP_EinfahrtPunkt

Definition:	From the XPlanung 5.2 specification: Punktförmig abgebildete Einfahrt (§9 Abs. 1 Nr. 11 und Abs. 6 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ der Einfahrt.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_EinfahrtTypen (enumeration)
Values	1000 **Einfahrt** 2000 **Ausfahrt** 3000 **EinAusfahrt**

1.5.18 BP_EinfahrtsbereichLinie

BP_EinfahrtsbereichLinie

Definition:	From the XPlanung 5.2 specification: Linienhaft modellierter Einfahrtsbereich (§9 Abs. 1 Nr. 11 und Abs. 6 BauGB). Durch die Digitalisierungsreihenfolge der Linienstützpunkte muss sichergestellt sein, dass die angrenzende Einfahrt relativ zur Laufrichtung auf der linken Seite liegt.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ der Einfahrt.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_EinfahrtTypen (enumeration)

Values

1000	**Einfahrt**
2000	**Ausfahrt**
3000	**EinAusfahrt**

1.5.19 BP_Eingriffsbereich

BP_Eingriffsbereich

Definition:	From the XPlanung 5.2 specification: Bestimmt einen Bereich, in dem ein Eingriff nach dem Naturschutzrecht zugelassen wird, der durch geeignete Flächen oder Maßnahmen ausgeglichen werden muss.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.20 BP_EmissionskontingentLaerm

BP_EmissionskontingentLaerm

Definition:	From the XPlanung 5.2 specification: Lärmemissionskontingent eines Teilgebietes nach DIN 45691, Abschnitt 4.6

Supertype of:	BP_EmissionskontingentLaermGebiet
Type:	Data type
Attribute:	
Name:	ekwertNacht
Definition:	From the XPlanung 5.2 specification: Emissionskontingent Nacht in db.
Voidable:	false
Multiplicity:	1
Value type:	Measure
Attribute:	
Name:	ekwertTag
Definition:	From the XPlanung 5.2 specification: Emissionskontingent Tag in db.
Voidable:	false
Multiplicity:	1
Value type:	Measure
Attribute:	
Name:	erlaeterung
Definition:	From the XPlanung 5.2 specification: Erläuterung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.21 BP_EmissionskontingentLaermGebiet

BP_EmissionskontingentLaermGebiet	
Definition:	From the XPlanung 5.2 specification: Lärmemissionskontingent eines Teilgebietes, das einem bestimmten Immissionsgebiet außerhalb des Geltungsbereiches des BPlans zugeordnet ist (Anhang A4 von DIN 45691).
Subtype of:	BP_EmissionskontingentLaerm
Type:	Data type
Attribute:	

Name:	gebietsbezeichnung
Definition:	From the XPlanung 5.2 specification: Bezeichnung des Immissionsgebietes.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.5.22 BP_ErhaltungsBereichFlaeche

BP_ErhaltungsBereichFlaeche

Definition:	From the XPlanung 5.2 specification: Fläche, auf denen der Rückbau, die Änderung oder die Nutzungsänderung baulichen Anlagen der Genehmigung durch die Gemeinde bedarf (§172 BauGB) Die Klasse wird als veraltet gekennzeichnet und fällt in XPlanGML V. 6.0 weg. Stattdessen sollte die Klasse SO_Gebiet verwendet werden.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	grund
Definition:	From the XPlanung 5.2 specification: Erhaltungsgrund.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_ErhaltungsGrund (enumeration)

Values

1000	**StaedtebaulicheGestalt**
2000	**Wohnbevoelkerung**
3000	**Umstrukturierung**

1.5.23 BP_FestsetzungNachLandesrecht

BP_FestsetzungNachLandesrecht

Definition:	From the XPlanung 5.2 specification: Festsetzung nach § 9 Nr. (4) BauGB.

Subtype of:	ZoningElement
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	kurzbeschreibung
Definition:	From the XPlanung 5.2 specification:
	Kurzbeschreibung der Festsetzung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.24 BP_FestsetzungenBaugebiet

BP_FestsetzungenBaugebiet	
Definition:	From the XPlanung 5.2 specification:
	Parameter zur Festsetzung des Maßes der baulichen Nutzung.
Supertype of:	BP_BaugebietsTeilFlaeche
	BP_BesondererNutzungszweckFlaeche
	BP_GemeinbedarfsFlaeche
	BP_GruenFlaeche
	BP_SpielSportanlagenFlaeche
	BP_StrassenVerkehrsFlaeche
	BP_UeberbaubareGrundstuecksFlaeche
	BP_VerEntsorgung
	BP_VerkehrsflaecheBesondererZweckbestimmung
Type:	Object type
Attribute:	
Name:	BM
Definition:	From the XPlanung 5.2 specification:
	Maximal zulässige Baumasse.
Voidable:	false
Multiplicity:	0..1
Value type:	Volume

Attribute:	
Name:	BMZ
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Baumassenzahl.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	BMZ_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Baumassenzahl.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	BM_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Baumasse.
Voidable:	false
Multiplicity:	0..1
Value type:	Volume
Attribute:	
Name:	Bmax
Definition:	From the XPlanung 5.2 specification: Maximale Breite von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Bmin
Definition:	From the XPlanung 5.2 specification: Minimale Breite von Baugrundstücken.

Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Fmax
Definition:	From the XPlanung 5.2 specification: Höchstmaß für die Größe (Fläche) eines Baugrundstücks.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	Fmin
Definition:	From the XPlanung 5.2 specification: Mindestmaß für die Größe (Fläche) eines Baugrundstücks.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GF
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Geschossfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GFZ
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Geschossflächenzahl.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	

Name:	GFZ_Ausn
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Geschossflächenzahl als Ausnahme.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GFZmax
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Geschossflächenzahl bei einer Bereichsangabe. Das Attribut GFZmin muss ebenfalls belegt sein.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GFZmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Geschossflächenzahl .
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GF_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Geschossfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GFmax
Definition:	From the XPlanung 5.2 specification:

	Maximal zulässige Geschossfläche bei einer Bereichsabgabe. Das Attribut GFmin muss ebenfalls belegt sein.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GFmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Geschossfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GR
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Grundfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GRZ
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Grundflächenzahl.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GRZ_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Grundflächenzahl.
Voidable:	false
Multiplicity:	0..1

Value type:	Decimal
Attribute:	
Name:	GRZmax
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Grundflächenzahl bei einer Bereichsangabe. Das Attribut GRZmin muss ebenfalls spezifiziert werden.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GRZmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Grundflächenzahl.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	GR_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Grundfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	GRmax
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Grundfläche bei einer Bereichsangabe. Das Attribut GRmin muss ebenfalls spezifiziert werden.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	

Name:	GRmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Grundfläche.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	Tmax
Definition:	From the XPlanung 5.2 specification: Maximale Tiefe von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Tmin
Definition:	From the XPlanung 5.2 specification: Minimale Tiefe von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Z
Definition:	From the XPlanung 5.2 specification: Maximalzahl der oberirdischen Vollgeschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	ZU
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Zahl der unterirdischen Geschosse.
Voidable:	false

Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	ZU_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Zahl der unterirdischen Geschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	ZUmax
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Zahl der unterirdischen Geschosse bei einer Bereichsangabe. Das Attribut ZUmin muss ebenfalls belegt sein.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	ZUmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Zahl der unterirdischen Geschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	ZUzwingend
Definition:	From the XPlanung 5.2 specification: Zwingend vorgeschriebene Zahl der unterirdischen Geschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	Z_Ausn
Definition:	From the XPlanung 5.2 specification: Ausnahmsweise maximal zulässige Zahl der oberirdischen Vollgeschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	Z_Dach
Definition:	From the XPlanung 5.2 specification: Maximalzahl der zusätzlich erlaubten Dachgeschosse, die gleichzeitig Vollgeschosse sind.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	Z_Staffel
Definition:	From the XPlanung 5.2 specification: Maximalzahl von oberirdischen zurückgesetzten Vollgeschossen als Staffelgeschoss..
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	Zmax
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Zahl der oberirdischen Vollgeschosse bei einer Bereichsangabe. Das Attribut Zmin muss ebenfalls belegt sein.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	Zmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Zahl der oberirdischen Vollgeschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	Zzwingend
Definition:	From the XPlanung 5.2 specification: Zwingend vorgeschriebene Zahl der oberirdischen Vollgeschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	maxZahlWohnungen
Definition:	From the XPlanung 5.2 specification: Höchstzulässige Zahl der Wohnungen in Wohngebäuden.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.5.25 BP_FirstRichtungsLinie

BP_FirstRichtungsLinie	
Definition:	From the XPlanung 5.2 specification: Gestaltungs-Festsetzung der Firstrichtung, beruhend auf Landesrecht, gemäß §9 Abs. 4 BauGB.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.26 BP_FoerderungsFlaeche

BP_FoerderungsFlaeche	
Definition:	From the XPlanung 5.2 specification:

	Fläche, auf der ganz oder teilweise nur Wohngebäude, die mit Mitteln der sozialen Wohnraumförderung gefördert werden könnten, errichtet werden dürfen (§9, Abs. 1, Nr. 7 BauGB).
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type

1.5.27 BP_FreiFlaeche

BP_FreiFlaeche	
Definition:	From the XPlanung 5.2 specification: Umgrenzung der Flächen, die von der Bebauung freizuhalten sind, und ihre Nutzung (§ 9 Abs. 1 Nr. 10 BauGB).
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	nutzung
Definition:	From the XPlanung 5.2 specification: Festgesetzte Nutzung der Freifläche.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.28 BP_GemeinbedarfsFlaeche

BP_GemeinbedarfsFlaeche	
Definition:	From the XPlanung 5.2 specification: Einrichtungen und Anlagen zur Versorgung mit Gütern und Dienstleistungen des öffentlichen und privaten Bereichs, hier Flächen für den Gemeindebedarf (§9, Abs. 1, Nr.5 und Abs. 6 BauGB).
Subtype of:	ZoningElement
	BP_Objekt
	BP_GestaltungBaugebiet
	BP_FestsetzungenBaugebiet
Type:	Feature type

Attribute:

Name: abweichendeBauweise

Definition: From the XPlanung 5.2 specification:

Nähere Bezeichnung einer "Abweichenden Bauweise" ("bauweise" == 3000).

Voidable: false

Multiplicity: 0..1

Value type: BP_AbweichendeBauweise (code list)

Attribute:

Name: bauweise

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_Bauweise (enumeration)

Values

1000	**OffeneBauweise**
2000	**GeschlosseneBauweise**
3000	**AbweichendeBauweise**

Attribute:

Name: bebauungsArt

Definition: From the XPlanung 5.2 specification:

Detaillierte Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Voidable: false

Multiplicity: 0..1

Value type: BP_BebauungsArt (enumeration)

Values

1000	**Einzelhaeuser**
2000	**Doppelhaeuser**
3000	**Hausgruppen**
4000	**EinzelDoppelhaeuser**
5000	**EinzelhaeuserHausgruppen**
6000	**DoppelhaeuserHausgruppen**
7000	**Reihenhaeuser**

	8000	**EinzelhaeuserDoppelhaeuserHausgruppen**

Attribute:

Name: detaillierteZweckbestimmung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere Festlegung der Zweckbestimmung.

Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".

Voidable: false

Multiplicity: 0..*

Value type: BP_DetailZweckbestGemeinbedarf (code list)

Attribute:

Name: zugunstenVon

Definition: From the XPlanung 5.2 specification:

Angabe des Begünstigten einer Ausweisung.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Zweckbestimmung der festgesetzten Fläche.

Voidable: false

Multiplicity: 0..*

Value type: XP_ZweckbestimmungGemeinbedarf (enumeration)

Values

1000	**OeffentlicheVerwaltung**
10000	**KommunaleEinrichtung**
10001	**BetriebOeffentlZweckbestimmung**
10002	**AnlageBundLand**
10003	**SonstigeOeffentlicheVerwaltung**
1200	**BildungForschung**
12000	**Schule**

12001	**Hochschule**
12002	**BerufsbildendeSchule**
12003	**Forschungseinrichtung**
12004	**SonstigesBildungForschung**
1400	**Kirche**
14000	**Sakralgebaeude**
14001	**KirchlicheVerwaltung**
14002	**Kirchengemeinde**
14003	**SonstigesKirche**
1600	**Sozial**
16000	**EinrichtungKinder**
16001	**EinrichtungJungendliche**
16002	**EinrichtungFamilienErwachsene**
16003	**EinrichtungSenioren**
16004	**SonstigeSozialeEinrichtung**
1800	**Gesundheit**
18000	**Krankenhaus**
18001	**SonstigesGesundheit**
2000	**Kultur**
20000	**MusikTheater**
20001	**Bildung**
20002	**SonstigeKultur**
2200	**Sport**
22000	**Bad**
22001	**SportplatzSporthalle**
22002	**SonstigerSport**
2400	**SicherheitOrdnung**
24000	**Feuerwehr**
24001	**Schutzbauwerk**
24002	**Justiz**
24003	**SonstigeSicherheitOrdnung**
2600	**Insfrastruktur**

	26000	**Post**
	26001	**SonstigeInfrastruktur**
	9999	**Sonstiges**

1.5.29 BP_GemeinschaftsanlagenFlaeche

BP_GemeinschaftsanlagenFlaeche	
Definition:	From the XPlanung 5.2 specification: Fläche für Gemeinschaftsanlagen für bestimmte räumliche Bereiche wie Kinderspielplätze, Freizeiteinrichtungen, Stellplätze und Garagen (§ 9 Abs. 1 Nr. 22 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	Zmax
Definition:	From the XPlanung 5.2 specification: Maximale Anzahl von Garagen-Geschossen.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codelist definierte detailliertere Festlegung der Zweckbestimmung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestGemeinschaftsanlagen (code list)
Association role	
Name:	eigentuemer
Voidable:	false
Multiplicity:	0..*

Value type: BP_BaugebietsTeilFlaeche (feature type)

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Zweckbestimmung der Fläche.

Voidable: false

Multiplicity: 0..*

Value type: BP_ZweckbestimmungGemeinschaftsanlagen (enumeration)

Values

1000	**Gemeinschaftsstellplaetze**
2000	**Gemeinschaftsgaragen**
3000	**Spielplatz**
3100	**Carport**
3200	**GemeinschaftsTiefgarage**
3300	**Nebengebaeude**
3400	**AbfallSammelanlagen**
3500	**EnergieVerteilungsanlagen**
3600	**AbfallWertstoffbehaelter**
3700	**Freizeiteinrichtungen**
3800	**Laermschutzanlagen**
3900	**AbwasserRegenwasser**
4000	**Ausgleichsmassnahmen**
9999	**Sonstiges**
4100	**Fahrradstellplaetze**
4200	**Gemeinschaftsdachgaerten**
4300	**GemeinschaftlichNutzbareDachflaechen**

1.5.30 BP_GemeinschaftsanlagenZuordnung

BP_GemeinschaftsanlagenZuordnung

Definition: From the XPlanung 5.2 specification:

Zuordnung von Gemeinschaftsanlagen zu Grundstücken.

Subtype of: SupplementaryRegulation

BP_Objekt

Type:	Feature type
Association role	
Name:	zuorndung
Voidable:	false
Multiplicity:	0..*
Value type:	BP_GemeinschaftsanlagenFlaeche (feature type)

1.5.31 BP_GestaltungBaugebiet

BP_GestaltungBaugebiet	
Definition:	From the XPlanung 5.2 specification: Parameter zur Festsetzung von Gestaltungs-Merkmalen.
Supertype of:	BP_BaugebietsTeilFlaeche BP_BesondererNutzungszweckFlaeche BP_GemeinbedarfsFlaeche BP_UeberbaubareGrundstuecksFlaeche
Type:	Object type
Attribute:	
Name:	DN
Definition:	From the XPlanung 5.2 specification: Maximal zulässige Dachneigung. Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	DNmax
Definition:	From the XPlanung 5.2 specification: Maxmal zulässige Dachneigung bei einer Bereichsangabe. Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.
Voidable:	false

Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	DNmin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Dachneigung bei einer Bereichsangabe. Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	DNzwingend
Definition:	From the XPlanung 5.2 specification: Zwingend vorgeschriebene Dachneigung. Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	FR
Definition:	From the XPlanung 5.2 specification: Vorgeschriebene Firstrichtung.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle
Attribute:	
Name:	dachform
Definition:	From the XPlanung 5.2 specification: Erlaubte Dachformen.

Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.

Voidable: false

Multiplicity: 0..*

Value type: BP_Dachform (enumeration)

Values

1000	**Flachdach**
2100	**Pultdach**
2200	**VersetztesPultdach**
3000	**GeneigtesDach**
3100	**Satteldach**
3200	**Walmdach**
3300	**Krueppelwalmdach**
3400	**Mansarddach**
3500	**Zeltdach**
3600	**Kegeldach**
3700	**Kuppeldach**
3800	**Sheddach**
3900	**Bogendach**
4000	**Turmdach**
4100	**Tonnendach**
5000	**Mischform**
9999	**Sonstiges**

Attribute:

Name: dachgestaltung

Definition: From the XPlanung 5.2 specification:

Parameter zur Einschränkung der zulässigen Dachformen.

Voidable: false

Multiplicity: 0..*

Value type: BP_Dachgestaltung (data type)

Attribute:

Name: detallierteDachform

Definition: From the XPlanung 5.2 specification:

	Über eine Codeliste definiertere detailliertere Dachform. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteDachform" bezieht sich auf den an gleicher Position stehenden Attributwert von dachform. Dies Attribut ist veraltet und wird in Version 6.0 wegfallen. Es sollte stattdessen der Datentyp BP_Dachgestaltung (Attribut dachgestaltung) verwendet werden.
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailDachform (code list)

1.5.32 BP_GewaesserFlaeche

BP_GewaesserFlaeche	
Definition:	From the XPlanung 5.2 specification: Festsetzung neuer Wasserflächen nach §9 Abs. 1 Nr. 16a BauGB. Diese Klasse wird in der nächsten Hauptversion des Standards eventuell wegfallen und durch SO_Gewaesser ersetzt werden.
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type
Attribute:	
Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Zweckbestimmung der Fläche.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_DetailZweckbestGewaesser (code list)
Attribute:	
Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmung der Wasserfläche.
Voidable:	false
Multiplicity:	0..1

Value type:	XP_ZweckbestimmungGewaesser (enumeration)
Values	1000 **Hafen** 10000 **Sportboothafen** 1100 **Wasserflaeche** 1200 **Fliessgewaesser** 9999 **Sonstiges**

1.5.33 BP_GruenFlaeche

BP_GruenFlaeche	
Definition:	From the XPlanung 5.2 specification: Festsetzungen von öffentlichen und privaten Grünflächen (§ 9, Abs. 1, Nr. 15 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt BP_FestsetzungenBaugebiet
Type:	Feature type
Attribute:	
Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Festlegung der Zweckbestimmung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestGruenFlaeche (code list)
Attribute:	
Name:	nutzungsform
Definition:	From the XPlanung 5.2 specification: Nutzungsform der festgesetzten Fläche.
Voidable:	false
Multiplicity:	0..1

Value type: XP_Nutzungsform (enumeration)

Values

1000	**Privat**
2000	**Oeffentlich**

Attribute:

Name: zugunstenVon

Definition: From the XPlanung 5.2 specification:

Angabe des Begünstigten einer Ausweisung.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Zweckbestimmung der Grünfläche.

Voidable: false

Multiplicity: 0..*

Value type: XP_ZweckbestimmungGruen (enumeration)

Values

1000	**Parkanlage**
10000	**ParkanlageHistorisch**
10001	**ParkanlageNaturnah**
10002	**ParkanlageWaldcharakter**
10003	**NaturnaheUferParkanlage**
1200	**Dauerkleingarten**
12000	**ErholungsGaerten**
1400	**Sportplatz**
14000	**Reitsportanlage**
14001	**Hundesportanlage**
14002	**Wassersportanlage**
14003	**Schiessstand**
14004	**Golfplatz**
14005	**Skisport**
14006	**Tennisanlage**

14007	**SonstigerSportplatz**	
1600	**Spielplatz**	
16000	**Bolzplatz**	
16001	**Abenteuerspielplatz**	
1800	**Zeltplatz**	
18000	**Campingplatz**	
2000	**Badeplatz**	
2200	**Freizeiterholung**	
22000	**Kleintierhaltung**	
22001	**Festplatz**	
2400	**SpezGruenflaeche**	
24000	**StrassenbegleitGruen**	
24001	**BoeschungsFlaeche**	
24002	**FeldWaldWiese**	
24003	**Uferschutzstreifen**	
24004	**Abschirmgruen**	
24005	**UmweltbildungsparkSchaugatter**	
24006	**RuhenderVerkehr**	
2600	**Friedhof**	
9999	**Sonstiges**	
99990	**Gaertnerei**	

1.5.34 BP_HoehenMass

BP_HoehenMass

Definition:	From the XPlanung 5.2 specification: Festsetzungen nach §9 Abs. 1 Nr. 1 BauGB für übereinanderliegende Geschosse und Ebenen und sonstige Teile baulicher Anlagen (§9 Abs.3 BauGB), sowie Hinweise auf Geländehöhen. Die Höhenwerte werden über das Attribut hoehenangabe der Basisklasse XP_Objekt spezifiziert.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.35 BP_Immissionsschutz

BP_Immissionsschutz

Definition:	From the XPlanung 5.2 specification: Festsetzung einer von der Bebauung freizuhaltenden Schutzfläche und ihre Nutzung, sowie einer Fläche für besondere Anlagen und Vorkehrungen zum Schutz vor schädlichen Umwelteinwirkungen und sonstigen Gefahren im Sinne des Bundes-Immissionsschutzgesetzes sowie die zum Schutz vor solchen Einwirkungen oder zur Vermeidung oder Minderung solcher Einwirkungen zu treffenden baulichen und sonstigen technischen Vorkehrungen (§9, Abs. 1, Nr. 24 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	detaillierteTechnVorkehrung
Definition:	From the XPlanung 5.2 specification: Detaillierte Klassifizierung der auf der Fläche zu treffenden baulichen oder sonstigen technischen Vorkehrungen.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_DetailTechnVorkehrungImmissionsschutz (code list)

Attribute:

Name:	laermpegelbereich
Definition:	From the XPlanung 5.2 specification: Festlegung der erforderlichen Luftschalldämmung von Außenbauteilen nach DIN 4109.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_Laermpegelbereich (enumeration)

Values

1000	**I**
1100	**II**
1200	**III**
1300	**IV**
1400	**V**

1500	**VI**
1600	**VII**

Attribute:

Name: nutzung

Definition: From the XPlanung 5.2 specification:

Festgesetzte Nutzung einer Schutzfläche.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: technVorkehrung

Definition: From the XPlanung 5.2 specification:

Klassifizierung der auf der Fläche zu treffenden baulichen oder sonstigen technischen Vorkehrungen.

Voidable: false

Multiplicity: 0..1

Value type: BP_TechnVorkehrungenImmissionsschutz (enumeration)

Values

1000	**Laermschutzvorkehrung**
10000	**FassadenMitSchallSchutzmassnahmen**
10001	**Laermschutzwand**
10002	**Laermschutzwall**
9999	**SonstigeVorkehrung**

Attribute:

Name: typ

Definition: From the XPlanung 5.2 specification:

Differenzierung der Immissionsschutz-Fläche.

Voidable: false

Multiplicity: 0..1

Value type: BP_ImmissionsschutzTypen (enumeration)

Values

1000	**Schutzflaeche**
2000	**BesondereAnlagenVorkehrungen**

1.5.36 BP_KennzeichnungsFlaeche

BP_KennzeichnungsFlaeche

Definition:	From the XPlanung 5.2 specification: Flächen für Kennzeichnungen gemäß §9 Abs. 5 BauGB.
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type

Attribute:

Name:	istVerdachtsflaeche
Definition:	From the XPlanung 5.2 specification: Legt fest, ob eine Altlast-Verdachtsfläche vorliegt.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean

Attribute:

Name:	nummer
Definition:	From the XPlanung 5.2 specification: Nummer im Altlastkataster.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmung der Kennzeichnungs-Fläche.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_ZweckbestimmungKennzeichnung (enumeration)

Values

1000	**Naturgewalten**
2000	**Abbauflaeche**
3000	**AeussereEinwirkungen**
4000	**SchadstoffBelastBoden**

5000	**LaermBelastung**	
6000	**Bergbau**	
7000	**Bodenordnung**	
8000	**Vorhabensgebiet**	
9999	**AndereGesetzlVorschriften**	

1.5.37 BP_KleintierhaltungFlaeche

BP_KleintierhaltungFlaeche

Definition:	From the XPlanung 5.2 specification: Fläche für die Errichtung von Anlagen für die Kleintierhaltung wie Ausstellungs- und Zuchtanlagen, Zwinger, Koppeln und dergleichen (§ 9 Abs. 1 Nr. 19 BauGB).
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type

1.5.38 BP_Landwirtschaft

BP_Landwirtschaft

Definition:	From the XPlanung 5.2 specification: Festsetzungen für die Landwirtschaft (§ 9, Abs. 1, Nr. 18a BauGB) Die Klasse wird als veraltet gekennzeichnet und wird in Version 6.0 wegfallen. Es sollte stattdessen die Klasse BP_LandwirtschaftsFlaeche verwendet werden.
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type

Attribute:

Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 Specification: Über eine Codeliste definierte detailliertere Zweckbestimmung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".

Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestLandwirtschaft (code list)

Attribute:

Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 Specification: Zweckbestimmungen der Ausweisung.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_ZweckbestimmungLandwirtschaft (enumeration)

Values

1000	**LandwirtschaftAllgemein**
1100	**Ackerbau**
1200	**WiesenWeidewirtschaft**
1300	**GartenabulicheErzeugung**
1400	**Obstbau**
1500	**Weinbau**
1600	**Imkerei**
1700	**Binnenfischerei**
9999	**Sonstiges**

1.5.39 BP_LandwirtschaftsFlaeche

BP_LandwirtschaftsFlaeche

Definition:	From the XPlanung 5.2 specification: Festsetzungen für die Landwirtschaft (§ 9, Abs. 1, Nr. 18a BauGB).
Subtype of:	ZoningElement BP_Objekt
Type:	Feature type

Attribute:

Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Zweckbestimmung.

	Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestLandwirtschaft (code list)

Attribute:

Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmungen der Ausweisung.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_ZweckbestimmungLandwirtschaft (enumeration)

Values

1000	**LandwirtschaftAllgemein**
1100	**Ackerbau**
1200	**WiesenWeidewirtschaft**
1300	**GartenabulicheErzeugung**
1400	**Obstbau**
1500	**Weinbau**
1600	**Imkerei**
1700	**Binnenfischerei**
9999	**Sonstiges**

1.5.40 BP_NebenanlagenAusschlussFlaeche

BP_NebenanlagenAusschlussFlaeche

Definition:	From the XPlanung 5.2 specification: Festsetzung einer Fläche für die Einschränkung oder den Ausschluss von Nebenanlagen nach §14 Absatz 1 Satz 3 BauNVO.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	typ
Definition:	From the XPlanung 5.2 specification: Art des Ausschlusses.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_NebenanlagenAusschlussTyp (enumeration)

Values

1000	**Einschraenkung**
2000	**Ausschluss**

1.5.41 BP_NebenanlagenFlaeche

BP_NebenanlagenFlaeche

Definition:	From the XPlanung 5.2 specification: Fläche für Nebenanlagen, die auf Grund anderer Vorschriften für die Nutzung von Grundstücken erforderlich sind, wie Spiel-, Freizeit- und Erholungsflächen sowie die Fläche für Stellplätze und Garagen mit ihren Einfahrten (§9 Abs. 1 Nr. 4 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	Zmax
Definition:	From the XPlanung 5.2 specification: Maximale Anzahl der Garagengeschosse.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	detallierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine CodeList definierte detailliertere Festlegung der Zweckbestimmung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".

Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestNebenanlagen (code list)

Attribute:

Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmung der Nebenanlagen-Fläche.
Voidable:	false
Multiplicity:	0..*
Value type:	BP_ZweckbestimmungNebenanlagen (enumeration)

Values

1000	**Stellplaetze**
2000	**Garagen**
3000	**Spielplatz**
3100	**Carport**
3200	**Tiefgarage**
3300	**Nebengebaeude**
3400	**AbfallSammelanlagen**
3500	**EnergieVerteilungsanlagen**
3600	**AbfallWertstoffbehaelter**
3700	**Fahrradstellplaetze**
9999	**Sonstiges**

1.5.42 BP_NichtUeberbaubareGrundstuecksflaeche

BP_NichtUeberbaubareGrundstuecksflaeche

Definition:	From the XPlanung 5.2 specification: Festlegung der nicht-überbaubaren Grundstücksfläche.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	nutzung
Definition:	From the XPlanung 5.2 specification:

Voidable:	false
Multiplicity:	0..1
Value type:	BP_NutzungNichtUeberbaubGrundstFlaeche (code list)

1.5.43 BP_NutzungsartenGrenze

BP_NutzungsartenGrenze

Definition:	From the XPlanung 5.2 specification: Abgrenzung unterschiedlicher Nutzung, z.B. von Baugebieten wenn diese nach PlanzVO in der gleichen Farbe dargestellt werden, oder Abgrenzung unterschiedlicher Nutzungsmaße innerhalb eines Baugebiets ("Knödellinie", § 1 Abs. 4, § 16 Abs. 5 BauNVO).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	detailTyp
Definition:	From the XPlanung 5.2 specification: Detaillierter Typ der Abgrenzung, wenn das Attribut typ den Wert 9999 (Sonstige Abgrenzung) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_DetailAbrenzungenTypen (code list)

Attribute:

Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ der Abgrenzung. Wenn das Attribut nicht belegt ist, ist die Abgrenzung eine Nutzungsarten-Grenze (Schlüsselnummer 1000).
Voidable:	false
Multiplicity:	0..1
Initial value:	1000
Value type:	BP_AbgrenzungenTypen (enumeration)
Values	1000 **Nutzungsartengrenze** 2000 **UnterschiedlicheHoehe**

	9999	**SonstigeAbgrenzung**	

1.5.44 BP_Objekt

BP_Objekt

Definition:	From the XPlanung 5.2 specification: Basisklasse für alle raumbezogenen Festsetzungen, Hinweise, Vermerke und Kennzeichnungen eines Bebauungsplans.
Subtype of:	XP_Objekt
Supertype of:	BP_AbgrabungsFlaeche BP_AbstandsFlaeche BP_AbstandsMass BP_AbweichungVonBaugrenze BP_AbweichungVonUeberbaubererGrundstuecksFlaeche BP_AnpflanzungBindungErhaltung BP_AufschuettungsFlaeche BP_AusgleichsFlaeche BP_AusgleichsMassnahme BP_BauGrenze BP_BauLinie BP_BaugebietsTeilFlaeche BP_BereichOhneEinAusfahrtLinie BP_BesondererNutzungszweckFlaeche BP_BodenschaetzeFlaeche BP_EinfahrtPunkt BP_EinfahrtsbereichLinie BP_Eingriffsbereich BP_ErhaltungsBereichFlaeche BP_FestsetzungNachLandesrecht BP_FirstRichtungsLinie BP_FoerderungsFlaeche BP_FreiFlaeche BP_GemeinbedarfsFlaeche

BP_GemeinschaftsanlagenFlaeche

BP_GemeinschaftsanlagenZuordnung

BP_GewaesserFlaeche

BP_GruenFlaeche

BP_HoehenMass

BP_Immissionsschutz

BP_KennzeichnungsFlaeche

BP_KleintierhaltungFlaeche

BP_Landwirtschaft

BP_LandwirtschaftsFlaeche

BP_NebenanlagenAusschlussFlaeche

BP_NebenanlagenFlaeche

BP_NichtUeberbaubareGrundstuecksflaeche

BP_NutzungsartenGrenze

BP_PersGruppenBestimmteFlaeche

BP_RegelungVergnuegungsstaetten

BP_RekultivierungsFlaeche

BP_RichtungssektorGrenze

BP_SchutzPflegeEntwicklungsFlaeche

BP_SchutzPflegeEntwicklungsMassnahme

BP_Sichtflaeche

BP_SpezielleBauweise

BP_SpielSportanlagenFlaeche

BP_StrassenVerkehrsFlaeche

BP_Strassenbegrenzungslinie

BP_Strassenkoerper

BP_TechnischeMassnahmenFlaeche

BP_TextlicheFestsetzungsFlaeche

BP_UeberbaubareGrundstuecksFlaeche

BP_UnverbindlicheVormerkung

BP_VerEntsorgung

BP_Veraenderungssperre

	BP_VerkehrsflaecheBesondererZweckbestimmung
	BP_WaldFlaeche
	BP_WasserwirtschaftsFlaeche
	BP_ZusatzkontingentLaerm
	BP_ZusatzkontingentLaermFlaeche
Type:	Feature type
Abstract:	true
Attribute:	
Name:	laermkontingent
Definition:	From the XPlanung 5.2 specification: Festsetzung eines Lärmemissionskontingent nach DIN 45691.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_EmissionskontingentLaerm (data type)
Attribute:	
Name:	laermkontingentGebiet
Definition:	From the XPlanung 5.2 specification: Festsetzung von Lärmemissionskontingenten nach DIN 45691, die einzelnen Immissionsgebieten zugeordnet sind.
Voidable:	false
Multiplicity:	0..*
Value type:	BP_EmissionskontingentLaermGebiet (data type)
Association role	
Name:	richtungssektorGrenze
Voidable:	false
Multiplicity:	0..*
Value type:	BP_RichtungssektorGrenze (feature type)
Association role	
Name:	wirdAusgeglichenDurchABE
Voidable:	false
Multiplicity:	0..*
Value type:	BP_AnpflanzungBindungErhaltung (feature type)

Association role	
Name:	wirdAusgeglichenDurchFlaeche
Voidable:	false
Multiplicity:	0..*
Value type:	BP_AusgleichsFlaeche (feature type)
Association role	
Name:	wirdAusgeglichenDurchMassnahme
Voidable:	false
Multiplicity:	0..*
Value type:	BP_AusgleichsMassnahme (feature type)
Association role	
Name:	wirdAusgeglichenDurchSPEFlaeche
Voidable:	false
Multiplicity:	0..*
Value type:	BP_SchutzPflegeEntwicklungsFlaeche (feature type)
Association role	
Name:	wirdAusgeglichenDurchSPEMassnahme
Voidable:	false
Multiplicity:	0..*
Value type:	BP_SchutzPflegeEntwicklungsMassnahme (feature type)
Association role	
Name:	zusatzkontingent
Voidable:	false
Multiplicity:	0..1
Value type:	BP_ZusatzkontingentLaerm (feature type)
Association role	
Name:	zusatzkontingentFlaeche
Voidable:	false
Multiplicity:	0..*
Value type:	BP_ZusatzkontingentLaermFlaeche (feature type)

1.5.45 BP_PersGruppenBestimmteFlaeche

BP_PersGruppenBestimmteFlaeche

Definition:	From the XPlanung 5.2 specification: Fläche, auf denen ganz oder teilweise nur Wohngebäude errichtet werden dürfen, die für Personengruppen mit besonderem Wohnbedarf bestimmt sind (§9, Abs. 1, Nr. 8 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.46 BP_RegelungVergnuegungsstaetten

BP_RegelungVergnuegungsstaetten

Definition:	From the XPlanung 5.2 specification: Festsetzung nach §9 Abs. 2b BauGB (Zulässigkeit von Vergnügungsstätten).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

Attribute:

Name:	zulaessigkeit
Definition:	From the XPlanung 5.2 specification: Zulässigkeit von Vergnügungsstätten.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_Zulaessigkeit (enumeration)

Values

1000	**Zulaessig**
2000	**NichtZulaessig**
3000	**AusnahmsweiseZulaessige**

1.5.47 BP_RekultivierungsFlaeche

BP_RekultivierungsFlaeche

Definition:	From the XPlanung 5.2 specification: Die Klasse wird als veraltet gekennzeichnet und wird in XPlanGML 6.0 wegfallen. Es sollte stattdessen die Klasse SO_SonstigesRecht verwendet werden.
Subtype of:	ZoningElement

	BP_Objekt
Type:	Feature type

1.5.48 BP_Richtungssektor

BP_Richtungssektor	
Definition:	From the XPlanung 5.2 specification:
Type:	Data type
Attribute:	
Name:	winkelAnfang
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	1
Value type:	Angle
Attribute:	
Name:	winkelEnde
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	1
Value type:	Angle
Attribute:	
Name:	zkWertNacht
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	1
Value type:	Measure
Attribute:	
Name:	zkWertTag
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	1
Value type:	Measure

1.5.49 BP_RichtungssektorGrenze

BP_RichtungssektorGrenze	
Definition:	From the XPlanung 5.2 specification: Linienhafte Repräsentation einer Richtungssektor-Grenze.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	winkel
Definition:	From the XPlanung 5.2 specification: Richtungswinkel der Sektorengrenze.
Voidable:	false
Multiplicity:	0..1
Value type:	Angle

1.5.50 BP_SchutzPflegeEntwicklungsFlaeche

BP_SchutzPflegeEntwicklungsFlaeche	
Definition:	From the XPlanung 5.2 specification: Maßnahmen zum Schutz, zur Pflege und zur Entwicklung von Natur und Landschaft (§9 Abs. 1 Nr. 20 und Abs. 4 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	istAugleich
Definition:	From the XPlanung 5.2 specification: Gibt an, ob die Maßnahme zum Ausgleich von Eingriffen genutzt wird.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean
Attribute:	
Name:	massnahme

Definition:	From the XPlanung 5.2 specification: Durchzuführende Maßnahme.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_SPEMassnahmenDaten (data type)

Attribute:

Name:	sonstZiel
Definition:	From the XPlanung 5.2 specification: Textlich formuliertes Ziel, wenn das Aztribut ziel den Wert 9999 (Sonstiges) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	ziel
Definition:	From the XPlanung 5.2 specification: Ziel der Maßnahme.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_SPEZiele (enumeration)

Values

1000	**SchutzPflege**
2000	**Entwicklung**
3000	**Anlage**
4000	**SchutzPflegeEntwicklung**
9999	**Sonstiges**

1.5.51 BP_SchutzPflegeEntwicklungsMassnahme

BP_SchutzPflegeEntwicklungsMassnahme

Definition:	From the XPlanung 5.2 specification: Maßnahmen zum Schutz, zur Pflege und zur Entwicklung von Natur und Landschaft (§9 Abs. 1 Nr. 20 und Abs. 4 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt

Type:	Feature type
Attribute:	
Name:	istAusgleich
Definition:	From the XPlanung 5.2 specification: Gibt an, ob die Maßnahme zum Ausgleich von Eingriffen genutzt wird.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean
Attribute:	
Name:	massnahme
Definition:	From the XPlanung 5.2 specification: Durchzuführende Maßnahme.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_SPEMassnahmenDaten (data type)
Attribute:	
Name:	sonstZiel
Definition:	From the XPlanung 5.2 specification: Textlich formuliertes Ziel, wenn das Aztribut ziel den Wert 9999 (Sonstiges) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	ziel
Definition:	From the XPlanung 5.2 specification: Ziel der Maßnahme.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_SPEZiele (enumeration)
Values	1000 **SchutzPflege**

2000	**Entwicklung**	
3000	**Anlage**	
4000	**SchutzPflegeEntwicklung**	
9999	**Sonstiges**	

1.5.52 BP_Sichtflaeche

BP_Sichtflaeche	
Definition:	From the XPlanung 5.2 specification:
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type

1.5.53 BP_SpezielleBauweise

BP_SpezielleBauweise	
Definition:	From the XPlanung 5.2 specification:
	Festsetzung der speziellen Bauweise / baulichen Besonderheit eines Gebäudes oder Bauwerks.
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	Bmax
Definition:	From the XPlanung 5.2 specification:
	Maximale Breite von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Bmin
Definition:	From the XPlanung 5.2 specification:
	Minimale Breite von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1

Value type:	Length
Attribute:	
Name:	Tmax
Definition:	From the XPlanung 5.2 specification: Maximale Tiefe von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	Tmin
Definition:	From the XPlanung 5.2 specification: Minimale Tiefe von Baugrundstücken.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	sonstTyp
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierter Typ der speziellen Bauweise, wenn typ den Wert 9999 (Sonstiges) hat.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_SpezielleBauweiseSonstTypen (code list)
Attribute:	
Name:	typ
Definition:	From the XPlanung 5.2 specification: Typ der speziellen Bauweise.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_SpezielleBauweiseTypen (enumeration)
Values	1000 **Durchfahrt** 1100 **Durchgang** 1200 **DurchfahrtDurchgang**

1300	**Auskragung**	
1400	**Arkade**	
1500	**Luftgeschoss**	
1600	**Bruecke**	
1700	**Tunnel**	
1800	**Rampe**	
9999	**Sonstiges**	

1.5.54 BP_SpielSportanlagenFlaeche

BP_SpielSportanlagenFlaeche

Definition:	From the XPlanung 5.2 specification: Einrichtungen und Anlagen zur Versorgung mit Gütern und Dienstleistungen des öffentlichen und privaten Bereichs, hier Flächen für Sport- und Spielanlagen (§9, Abs. 1, Nr. 5 und Abs. 6 BauGB).
Subtype of:	ZoningElement BP_Objekt BP_FestsetzungenBaugebiet
Type:	Feature type

Attribute:

Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Festlegung der Zweckbestimmung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestSpielSportanlage (code list)

Attribute:

Name:	zugunstenVon
Definition:	From the XPlanung 5.2 specification: Angabe des Begünstigten einer Ausweisung.

Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmung der festgesetzten Fläche.
Voidable:	false
Multiplicity:	0..*
Value type:	BP_SpielSportanlagenFlaeche (feature type)

1.5.55 BP_StrassenVerkehrsFlaeche

BP_StrassenVerkehrsFlaeche	
Definition:	From the XPlanung 5.2 specification: Strassenverkehrsfläche (§ 9 Abs. 1 Nr. 11 und Abs. 6 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt BP_FestsetzungenBaugebiet
Type:	Feature type
Association role	
Name:	begrenzungslinie
Voidable:	false
Multiplicity:	0..*
Value type:	BP_Strassenbegrenzungslinie (feature type)
Attribute:	
Name:	nutzungsform
Definition:	From the XPlanung 5.2 specification: Nutzungsform der Fläche.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_Nutzungsform (enumeration)
Values	1000 **Privat**

	2000	**Oeffentlich**	

1.5.56 BP_Strassenbegrenzungslinie

BP_Strassenbegrenzungslinie	
Definition:	From the XPlanung 5.2 specification: Straßenbegrenzungslinie (§ 9 Abs. 1 Nr. 11 und Abs. 6 BauGB) . Durch die Digitalisierungsreihenfolge der Linienstützpunkte muss sichergestellt sein, dass die abzugrenzende Straßenfläche relativ zur Laufrichtung auf der linken Seite liegt.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	bautiefe
Definition:	From the XPlanung 5.2 specification: Minimaler Abstand der Bebauung von der Straßenbegrenzungslinie.
Voidable:	false
Multiplicity:	0..1
Value type:	Length

1.5.57 BP_Strassenkoerper

BP_Strassenkoerper	
Definition:	From the XPlanung 5.2 specification: Flächen für Aufschüttungen, Abgrabungen und Stützmauern, soweit sie zur Herstellung des Straßenkörpers erforderlich sind (§9, Abs. 1, Nr. 26 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	typ
Definition:	From the XPlanung 5.2 specification:

	Notwendige Maßnahme zur Herstellung des Straßenkörpers.
Voidable:	false
Multiplicity:	1
Value type:	BP_StrassenkoerperHerstellung (enumeration)
Values	1000 **Aufschuettung** 2000 **Abgrabung** 3000 **Stuetzmauer**

1.5.58 BP_TechnischeMassnahmenFlaeche

BP_TechnischeMassnahmenFlaeche	
Definition:	From the XPlanung 5.2 specification: Fläche für technische oder bauliche Maßnahmen nach § 9, Abs. 1, Nr. 23 BauGB.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	technischeMassnahme
Definition:	From the XPlanung 5.2 specification: Beschreibung der Maßnahme.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Klassifikation der durchzuführenden Maßnahmen nach §9, Abs. 1, Nr. 23a BauGB.
Voidable:	false
Multiplicity:	1
Value type:	BP_ZweckbestimmungenTMF (enumeration)
Values	1000 **Luftreinhaltung** 2000 **NutzungErneuerbareEnergien**

	3000	**MinderungStoerfallfolgen**	

1.5.59 BP_TextlicheFestsetzungsFlaeche

BP_TextlicheFestsetzungsFlaeche

Definition:	From the XPlanung 5.2 specification: Bereich, in dem bestimmte Textliche Festsetzungen gültig sind, die über die Relation "refTextInhalt" (Basisklasse BP_Objekt) spezifiziert werden.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type

1.5.60 BP_UeberbaubareGrundstuecksFlaeche

BP_UeberbaubareGrundstuecksFlaeche

Definition:	From the XPlanung 5.2 specification: Festsetzung der überbaubaren Grundstücksfläche (§9, Abs. 1, Nr. 2 BauGB). Über die Attribute geschossMin und geschossMax kann die Festsetzung auf einen Bereich von Geschossen beschränkt werden. Wenn eine Einschränkung der Festsetzung durch expliziter Höhenangaben erfolgen soll, ist dazu die Oberklassen-Relation hoehenangabe auf den komplexen Datentyp XP_Hoehenangabe zu verwenden.
Subtype of:	SupplementaryRegulation BP_Objekt BP_GestaltungBaugebiet BP_ZusaetzlicheFestsetzungen BP_FestsetzungenBaugebiet
Type:	Feature type

Attribute:

Name:	abweichendeBauweise
Definition:	From the XPlanung 5.2 specification: Nähere Bezeichnung einer "Abweichenden Bauweise" ("bauweise == 3000"). Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.
Voidable:	false

Multiplicity: 0..1

Value type: BP_AbweichendeBauweise (code list)

Association role

Name: baugrenze

Voidable: false

Multiplicity: 0..*

Value type: BP_BauGrenze (feature type)

Association role

Name: baulinie

Voidable: false

Multiplicity: 0..*

Value type: BP_BauLinie (feature type)

Attribute:

Name: bauweise

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: BP_Bauweise (enumeration)

Values

1000	**OffeneBauweise**
2000	**GeschlosseneBauweise**
3000	**AbweichendeBauweise**

Attribute:

Name: bebauungRueckwaertigeGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der rückwärtigen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungSeitlicheGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der seitlichen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungVordereGrenze

Definition: From the XPlanung 5.2 specification:

Festsetzung der Bebauung der vorderen Grundstücksgrenze (§9, Abs. 1, Nr. 2 BauGB).

Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: BP_GrenzBebauung (enumeration)

Values

1000	**Verboten**
2000	**Erlaubt**
3000	**Erzwungen**

Attribute:

Name: bebauungsArt

Definition: From the XPlanung 5.2 specification:

Detaillierte Festsetzung der Bauweise (§9, Abs. 1, Nr. 2 BauGB).

Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: BP_BebauungsArt (enumeration)

Values

1000	**Einzelhaeuser**
2000	**Doppelhaeuser**
3000	**Hausgruppen**
4000	**EinzelDoppelhaeuser**
5000	**EinzelhaeuserHausgruppen**
6000	**DoppelhaeuserHausgruppen**
7000	**Reihenhaeuser**
8000	**EinzelhaeuserDoppelhaeuserHausgruppen**

Attribute:

Name: geschossMax

Definition: From the XPlanung 5.2 specification:

Gibt bei geschossweiser Festsetzung die Nummer des Geschosses an, bis zu der die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse ab einschl. "geschossMin".

Voidable: false

Multiplicity: 0..1

Value type: Integer

Attribute:

Name: geschossMin

Definition: From the XPlanung 5.2 specification:

Gibt bei geschossweiser Festsetzung die Nummer des Geschosses an, ab den die Festsetzung gilt. Wenn das Attribut nicht belegt ist, gilt die Festsetzung für alle Geschosse bis einschl. "geschossMax".

Voidable: false

Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	vertikaleDifferenzierung
Definition:	From the XPlanung 5.2 specification: Gibt an, ob eine vertikale Differenzierung der Gebäude vorgeschrieben ist. Dieser Wert hat Priorität gegenüber einer im umschließenden Baugebiet (BP_BaugebietsTeilFlaeche) getroffenen Festsetzung.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean

1.5.61 BP_UnverbindlicheVormerkung

BP_UnverbindlicheVormerkung	
Definition:	From the XPlanung 5.2 specification: Unverbindliche Vormerkung späterer Planungsabsichten.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	vormerkung
Definition:	From the XPlanung 5.2 specification: Text der Vormerkung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.62 BP_VerEntsorgung

BP_VerEntsorgung	
Definition:	From the XPlanung 5.2 Specification:

	Flächen und Leitungen für Versorgungsanlagen, für die Abfallentsorgung und Abwasserbeseitigung sowie für Ablagerungen (§9 Abs. 1, Nr. 12, 14 und Abs. 6 BauGB).
Subtype of:	ZoningElement
	BP_Objekt
	BP_FestsetzungenBaugebiet
Type:	Feature type
Attribute:	
Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 Specification: Über eine Codeliste definierte detailliertere Zweckbestimmung der Festsetzung. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestVerEntsorgung (code list)
Attribute:	
Name:	textlicheErgaenzung
Definition:	From the XPlanung 5.2 Specification: Zusätzliche textliche Beschreibung der Ver- bzw. Entsorgungseinrichtung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zugunstenVon
Definition:	From the XPlanung 5.2 Specification: Angabe des Begünstigten der Ausweisung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	

Name: zweckbestimmung

Definition: From the XPlanung 5.2 Specification:

Zweckbestimmung der Festsetzung.

Voidable: false

Multiplicity: 0..*

Value type: XP_ZweckbestimmungVerEntsorgung (enumeration)

Values

1000	**Elektrizitaet**
10000	**Hochspannungsleitung**
10001	**TrafostationUmspannwerk**
10002	**Solarkraftwerk**
10003	**Windkraftwerk**
10004	**Geothermiekraftwerk**
10005	**Elektrizitaetswerk**
10006	**Wasserkraftwerk**
10007	**BiomasseKrafwerk**
10008	**Kabelleitung**
10009	**Niederspannungsleitung**
10010	**Leitungsmast**
1200	**Gas**
12000	**Ferngasleitung**
12001	**Gaswerk**
12002	**Gasbehaelter**
12003	**Gasdruckregler**
12004	**Gasstation**
12005	**Gasleitung**
1300	**Erdoel**
13000	**Erdoelleitung**
13001	**Bohrstelle**
13002	**Erdoelpumpstation**
13003	**Oeltank**
1400	**Waermeversorgung**
14000	**Blockheizkraftwerk**

14001	**Fernwaermeleitung**
14002	**Fernheizwerk**
1600	**Wasser**
16000	**Wasserwerk**
16001	**Wasserleitung**
16002	**Wasserspeicher**
16003	**Brunnen**
16004	**Pumpwerk**
16005	**Quelle**
1800	**Abwasser**
18000	**Abwasserleitung**
18001	**Abwasserrueckhaltebecken**
18002	**Abwasserpumpwerk**
18003	**Klaeranlage**
18004	**AnlageKlaerschlamm**
18005	**SonstigeAbwasserBehandlungsanlage**
18006	**SalzOderSoleleitungen**
2000	**Regenwasser**
20000	**RegenwaserRueckhaltebecken**
20001	**Niederschlagswasserleitungen**
2200	**Abfallentsorgung**
22000	**Muellumladestation**
22001	**Muellbeseitigungsanlage**
22002	**Muellsortieranlage**
22003	**Recyclinghof**
2400	**Ablagerung**
24000	**Erdaushubdeponie**
24001	**Bauschuttdeponie**
24002	**Hausmuelldeponie**
24003	**Sondermuelldeponie**
24004	**StillgelegteDeponie**
24005	**RekultivierteDeponie**

	2600	**Telekommunikation**
	26000	**Fernmeldeanlage**
	26001	**Mobilfunkanlage**
	26002	**Fernmeldekabel**
	2800	**ErneuerbareEnergien**
	3000	**KraftWaermeKopplung**
	9999	**Sonstiges**
	999990	**Produktenleitung**

1.5.63 BP_Veraenderungssperre

BP_Veraenderungssperre	
Definition:	From the XPlanung 5.2 specification: Ausweisung einer Veränderungssperre, die nicht den gesamten Geltungsbereich des Plans umfasst. Bei Verwendung dieser Klasse muss das Attribut "veraenderungssperre" des zugehörigen Plans (Klasse BP_Plan) auf "false" gesetzt werden.
Subtype of:	SupplementaryRegulation BP_Objekt
Type:	Feature type
Attribute:	
Name:	gueltigkeitsDatum
Definition:	From the XPlanung 5.2 specification: Datum, bis zu dem die Veränderungssperre bestehen bleibt.
Voidable:	false
Multiplicity:	1
Value type:	Date
Attribute:	
Name:	verlaengerung
Definition:	From the XPlanung 5.2 specification: Gibt an, ob die Veränderungssperre bereits ein- oder zweimal verlängert wurde.
Voidable:	false
Multiplicity:	1

Value type:	XP_VerlaengerungVeraenderungssperre (enumeration)
Values	1000 **Keine** 2000 **ErsteVerlaengerung** 3000 **ZweiteVerlaengerung**

1.5.64 BP_VerkehrsflaecheBesondererZweckbestimmung

BP_VerkehrsflaecheBesondererZweckbestimmung	
Definition:	From the XPlanung 5.2 specification: Verkehrsfläche besonderer Zweckbestimmung (§ 9 Abs. 1 Nr. 11 und Abs. 6 BauGB).
Subtype of:	SupplementaryRegulation BP_Objekt BP_FestsetzungenBaugebiet
Type:	Feature type
Association role	
Name:	begrenzungslinie
Voidable:	false
Multiplicity:	0..*
Value type:	BP_Strassenbegrenzungslinie (feature type)
Attribute:	
Name:	detaillierteZweckbestimmung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Zweckbestimmung der Fläche. Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".
Voidable:	false
Multiplicity:	0..*
Value type:	BP_DetailZweckbestStrassenverkehr (code list)
Attribute:	
Name:	nutzungsform
Definition:	From the XPlanung 5.2 specification:

Nutzungsform der Fläche.

Voidable: false

Multiplicity: 0..1

Value type: XP_Nutzungsform (enumeration)

Values

1000	**Privat**
2000	**Oeffentlich**

Attribute:

Name: zugunstenVon

Definition: From the XPlanung 5.2 specification:

Begünstigter der Festsetzung.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Zweckbestimmung der Fläche.

Voidable: false

Multiplicity: 0..*

Value type: BP_ZweckbestimmungStrassenverkehr (enumeration)

Values

1000	**Parkierungsflaeche**
1100	**Fussgaengerbereich**
1200	**VerkehrsberuhigterBereich**
1300	**RadGehweg**
1400	**Radweg**
1500	**Gehweg**
1550	**Wanderweg**
1560	**ReitKutschweg**
1580	**Wirtschaftsweg**
1600	**FahrradAbstellplatz**
1700	**UeberfuehrenderVerkehrsweg**
1800	**UnterfuehrenderVerkehrsweg**

2000	**P_RAnlage**
2100	**Platz**
2200	**Anschlussflaeche**
2300	**LandwirtschaftlicherVerkehr**
2400	**Verkehrsgruen**
2500	**Rastanlage**
2600	**Busbahnhof**
3000	**CarSharing**
3100	**BikeSharing**
3200	**B_RAnlage**
3300	**Parkhaus**
3400	**Mischverkehrsflaeche**
3500	**Ladestation**
9999	**Sonstiges**

1.5.65 BP_WaldFlaeche

BP_WaldFlaeche

Definition: From the XPlanung 5.2 specification:

Festsetzung von Waldflächen (§ 9, Abs. 1, Nr. 18b BauGB).

Subtype of: ZoningElement

BP_Objekt

Type: Feature type

Attribute:

Name: betreten

Definition: From the XPlanung 5.2 specification:

Festlegung zusätzlicher, normalerweise nicht-gestatteter Aktivitäten, die in dem Wald ausgeführt werden dürfen, nach §14 Abs. 2 Bundeswaldgesetz.

Voidable: false

Multiplicity: 0..*

Value type: XP_WaldbetretungTyp (enumeration)

Values

1000	**Radfahren**
2000	**Reiten**

3000	**Fahren**
4000	**Hundesport**

Attribute:

Name: detaillierteZweckbestimmung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere Festlegung der Funktion des Waldes.

Der an einer bestimmten Listenposition aufgeführte Wert von "detaillierteZweckbestimmung" bezieht sich auf den an gleicher Position stehenden Attributwert von "zweckbestimmung".

Voidable: false

Multiplicity: 0..*

Value type: BP_DetailZweckbestWaldFlaeche (code list)

Attribute:

Name: eigentumsart

Definition: From the XPlanung 5.2 specification:

Festlegung der Eigentumsart des Waldes.

Voidable: false

Multiplicity: 0..1

Value type: XP_EigentumsartWald (enumeration)

Values

1000	**OeffentlicherWald**
1100	**Staatswald**
1200	**Koerperschaftswald**
12000	**Kommunalwald**
12001	**Stiftungswald**
2000	**Privatwald**
20000	**Gemeinschaftswald**
20001	**Genossenschaftswald**
3000	**Kirchenwald**
9999	**Sonstiges**

Attribute:

Name: zweckbestimmung

Definition: From the XPlanung 5.2 specification:

Funktion der Waldfläche.

Voidable: false

Multiplicity: 0..*

Value type: XP_ZweckbestimmungWald (enumeration)

Values

1000	**Naturwald**
10000	**Waldschutzgebiet**
1200	**Nutzwald**
1400	**Erholungswald**
1600	**Schutzwald**
16000	**Bodenschutzwald**
16001	**Biotopschutzwald**
16002	**NaturnaherWald**
16003	**SchutzwaldSchaedlicheUmwelteinwirkungen**
16004	**Schonwald**
1700	**Bannwald**
1800	**FlaecheForstwirtschaft**
1900	**ImissionsgeschaedigterWald**
9999	**Sonstiges**

1.5.66 BP_WasserwirtschaftsFlaeche

BP_WasserwirtschaftsFlaeche

Definition: From the XPlanung 5.2 specification:

Flächen für die Wasserwirtschaft (§9 Abs. 1 Nr. 16a BauGB), sowie Flächen für Hochwasserschutz-anlagen und für die Regelung des Wasserabflusses (§9 Abs. 1 Nr. 16b BauGB).

Subtype of: ZoningElement

BP_Objekt

Type: Feature type

Attribute:

Name: detaillierteZweckbestimmung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere Zweckbestimmung der Fläche.

Voidable:	false
Multiplicity:	0..1
Value type:	BP_DetailZweckbestWasserwirtschaft (code list)

Attribute:

Name:	zweckbestimmung
Definition:	From the XPlanung 5.2 specification: Zweckbestimmung der Fläche.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_ZweckbestimmungWasserwirtschaft (enumeration)

Values

1000	**HochwasserRueckhaltebecken**
1100	**Ueberschwemmgebiet**
1200	**Versickerungsflaeche**
1300	**Entwaesserungsgraben**
1400	**Deich**
1500	**RegenRueckhaltebecken**
9999	**Sonstiges**

1.5.67 BP_ZusaetzlicheFestsetzungen

BP_ZusaetzlicheFestsetzungen

Supertype of:	BP_BaugebietsTeilFlaeche BP_UeberbaubareGrundstuecksFlaeche
Type:	Object type

Attribute:

Name:	GFAntGewerbe
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §6a Abs. (4) Nr. 4 BauNVO: Für urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden ein im Bebauungsplan bestimmter Anteil der zulässigen Geschossfläche für gewerbliche Nutzungen zu verwenden ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Scale

Attribute:

Name:	GFAntWohnen
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §4a Abs. (4) Nr. 2 bzw. §6a Abs. (4) Nr. 3 BauNVO: Für besondere Wohngebiete und urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden ein im Bebauungsplan bestimmter Anteil der zulässigen Geschossfläche für Wohnungen zu verwenden ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Scale

Attribute:

Name:	GFGewerbe
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §6a Abs. (4) Nr. 4 BauNVO: Für urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden eine im Bebauungsplan bestimmte Größe der Geschossfläche für gewerbliche Nutzungen zu verwenden ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Area

Attribute:

Name:	GFWohnen
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §4a Abs. (4) Nr. 2 bzw. §6a Abs. (4) Nr. 3 BauNVO: Für besondere Wohngebiete und urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden eine im Bebauungsplan bestimmte Größe der Geschossfläche für Wohnungen zu verwenden ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Area

Attribute:

Name:	VF

Definition:	From the XPlanung 5.2 specification: Festsetzung der maximal zulässigen Verkaufsfläche in einem Sondergebiet.
Voidable:	false
Multiplicity:	0..1
Value type:	Area

Attribute:

Name:	ZWohn
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §4a Abs. (4) Nr. 1 bzw. nach §6a Abs. (4) Nr. 2 BauNVO: Für besondere Wohngebiete und urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden oberhalb eines im Bebauungsplan bestimmten Geschosses nur Wohnungen zulässig sind.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	wohnnutzungEGStrasse
Definition:	From the XPlanung 5.2 specification: Festsetzung nach §6a Abs. (4) Nr. 1 BauNVO: Für urbane Gebiete oder Teile solcher Gebiete kann festgesetzt werden, dass in Gebäuden im Erdgeschoss an der Straßenseite eine Wohnnutzung nicht oder nur ausnahmsweise zulässig ist.
Voidable:	false
Multiplicity:	0..1
Value type:	BP_Zulaessigkeit (enumeration)
Values	1000 **Zulaessig** 2000 **NichtZulaessig** 3000 **AusnahmsweiseZulaessige**

1.5.68 BP_ZusatzkontingentLaerm

BP_ZusatzkontingentLaerm

Definition:	From the XPlanung 5.2 specification:

	Parametrische Spezifikation von zusätzlichen Lärmemissionskontingenten für einzelne Richtungssektoren (DIN 45691, Anhang 2).
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	bezeichnung
Definition:	From the XPlanung 5.2 specification:
	Bezeichnung des Kontingentes.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	richtungssektor
Definition:	From the XPlanung 5.2 specification:
	Spezifikation der Richtungssektoren.
Voidable:	false
Multiplicity:	1..*
Value type:	BP_Richtungssektor (data type)

1.5.69 BP_ZusatzkontingentLaermFlaeche

BP_ZusatzkontingentLaermFlaeche	
Definition:	From the XPlanung 5.2 specification:
	Bezeichnung des Kontingentes.
Subtype of:	SupplementaryRegulation
	BP_Objekt
Type:	Feature type
Attribute:	
Name:	bezeichnung
Definition:	From the XPlanung 5.2 specification:
Voidable:	false
Multiplicity:	0..1

Value type:	CharacterString
Attribute:	
Name:	richtungssektor
Definition:	From the XPlanung 5.2 specification: Spezifikation des zugehörigen Richtungssektors.
Voidable:	false
Multiplicity:	1
Value type:	BP_Richtungssektor (data type)

1.5.70 FP_Objekt

FP_Objekt	
Definition:	From the XPlanung 5.2 specification: Basisklasse für alle Fachobjekte des Flächennutzungsplans.
Subtype of:	XP_Objekt
Type:	Feature type
Abstract:	true

1.5.71 RP_Objekt

RP_Objekt	
Definition:	From the XPlanung 5.2 specification: RP_Objekt ist die Basisklasse für alle spezifischen Festlegungen eines Raumordnungsplans. Sie selbst ist abstrakt, d.h. sie wird selbst nicht als eigenes Objekt verwendet, sondern vererbt nur ihre Attribute an spezielle Klassen.
Subtype of:	XP_Objekt
Supertype of:	RP_Wasserschutz
Type:	Feature type
Abstract:	true
Attribute:	
Name:	bedeutsamkeit
Definition:	From the XPlanung 5.2 specification: Bedeutsamkeit eines Objekts.
Voidable:	false

Multiplicity: 0..*

Value type: RP_Bedeutsamkeit (enumeration)

Values

1000	**Regional**
2000	**Ueberregional**
3000	**Grossraeumig**
4000	**Landesweit**
5000	**Bundesweit**
6000	**Europaeisch**
7000	**International**
8000	**Flaechenschliessend**
9000	**Herausragend**

Attribute:

Name: gebietsTyp

Definition: From the XPlanung 5.2 specification:

Gebietstyp eines Objekts.

Voidable: false

Multiplicity: 0..*

Value type: RP_Gebietstyp (enumeration)

Values

1000	**Vorranggebiet**
1001	**Vorrangstandort**
1100	**Vorbehaltsgebiet**
1101	**Vorbehaltsstandort**
1200	**Eignungsgebiet**
1300	**VorrangundEignungsgebiet**
1400	**Ausschlussgebiet**
1500	**Vorsorgegebiet**
1501	**Vorsorgestandort**
1600	**Vorzugsraum**
1700	**Potenzialgebiet**
1800	**Schwerpunktraum**
9999	**SonstigesGebiet**

Attribute:

Name:	istZweckbindung
Definition:	From the XPlanung 5.2 specification: Zeigt an, ob es sich bei diesem Objekt um eine Zweckbindung handelt.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean
Attribute:	
Name:	konkretisierung
Definition:	From the XPlanung 5.2 specification: Konkretisierung des Rechtscharakters.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	kuestenmeer
Definition:	From the XPlanung 5.2 specification: Zeigt an, ob das Objekt im Küstenmeer liegt.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean

1.5.72 RP_Wasserschutz

RP_Wasserschutz	
Definition:	From the XPlanung 5.2 specification: Grund-, Trink- und Oberflächenwasserschutz.
Subtype of:	ZoningElement RP_Objekt
Type:	Feature type
Attribute:	
Name:	typ
Definition:	From the XPlanung 5.2 specification:

Klassifikation des Wasserschutztyps.

Voidable: false

Multiplicity: 0..1

Value type: RP_WasserschutzTypen (enumeration)

Values

1000	**Wasserschutzgebiet**
2000	**Grundwasserschutz**
2001	**Grundwasservorkommen**
2002	**Gewaesserschutz**
3000	**Trinkwasserschutz**
4000	**Trinkwassergewinnung**
5000	**Oberflaechenwasserschutz**
6000	**Heilquelle**
7000	**Wasserversorgung**
9999	**SonstigerWasserschutz**

Attribute:

Name: zone

Definition: From the XPlanung 5.2 specification:

Wasserschutzzone.

Voidable: false

Multiplicity: 0..*

Value type: RP_WasserschutzZonen (enumeration)

Values

1000	**Zone1**
2000	**Zone2**
3000	**Zone3**

1.5.73 SO_Denkmalschutzrecht

SO_Denkmalschutzrecht

Definition: From the XPlanung 5.2 specification:

Festlegung nach Denkmalschutzrecht.

Subtype of: SupplementaryRegulation

SO_Objekt

Type: Feature type

Attribute:

Name: artDerFestlegung

Definition: From the XPlanung 5.2 specification:

Rechtliche Klassifizierung der Festlegung.

Voidable: false

Multiplicity: 0..1

Value type: SO_KlassifizNachDenkmalschutzrecht (enumeration)

Values

1000	**DenkmalschutzEnsemble**
1100	**DenkmalschutzEinzelanlage**
1200	**Grabungsschutzgebiet**
1300	**PufferzoneWeltkulturerbeEnger**
1400	**PufferzoneWeltkulturerbeWeiter**
1500	**ArchaeologischesDenkmal**
1600	**Bodendenkmal**
9999	**Sonstiges**

Attribute:

Name: detailArtDerFestlegung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere rechtliche Klassifizierung der Festlegung.

Voidable: false

Multiplicity: 0..1

Value type: SO_DetailKlassifizNachDenkmalschutzrecht (code list)

Attribute:

Name: name

Definition: From the XPlanung 5.2 specification:

Informelle Bezeichnung der Festlegung.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: nummer

Definition: From the XPlanung 5.2 specification:

	Amtliche Bezeichnung / Kennziffer der Festlegung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	weltkulturerbe
Definition:	From the XPlanung 5.2 specification: Gibt an, ob das geschützte Objekt zum Weltkulturerbe gehört.
Voidable:	false
Multiplicity:	0..1
Initial value:	false
Value type:	Boolean

1.5.74 SO_Gebiet

SO_Gebiet	
Definition:	From the XPlanung 5.2 specification: Umgrenzung eines sonstigen Gebietes nach BauGB.
Subtype of:	SupplementaryRegulation SO_Objekt
Type:	Feature type
Attribute:	
Name:	aufstellungsbeschlussDatum
Definition:	From the XPlanung 5.2 specification: Datum des Aufstellungsbeschlusses.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	durchfuehrungEndDatum
Definition:	From the XPlanung 5.2 specification: End-Datum der Durchführung.
Voidable:	false

Multiplicity: 0..1

Value type: Date

Attribute:

Name: durchfuehrungStartDatum

Definition: From the XPlanung 5.2 specification:

Start-Datum der Durchführung.

Voidable: false

Multiplicity: 0..1

Value type: Date

Attribute:

Name: gebietsArt

Definition: From the XPlanung 5.2 specification:

Klassifikation des Gebietes nach BauGB.

Voidable: false

Multiplicity: 0..1

Value type: SO_GebietsArt (enumeration)

Values

1000	**Umlegungsgebiet**
1100	**StaedtebaulicheSanierung**
1200	**StaedtebaulicheEntwicklungsmassnahme**
1300	**Stadtumbaugebiet**
1400	**SozialeStadt**
1500	**BusinessImprovementDistrict**
1600	**HousingImprovementDistrict**
1999	**Erhaltungsverordnung**
2000	**ErhaltungsverordnungStaedtebaulicheGestalt**
2100	**ErhaltungsverordnungWohnbevoelkerung**
2200	**ErhaltungsverordnungUmstrukturierung**
9999	**Sonstiges**

Attribute:

Name: gemeinde

Definition: From the XPlanung 5.2 specification:

Zuständige Gemeinde.

Voidable: false

Multiplicity: 0..1

Value type: Municipality (feature type)

Attribute:

Name: rechtsstandGebiet

Definition: From the XPlanung 5.2 specification:

Rechtsstand der Gebietsausweisung.

Voidable: false

Multiplicity: 0..1

Value type: SO_RechtsstandGebietTyp (enumeration)

Values

1000	**VorbereitendeUntersuchung**
2000	**Aufstellung**
3000	**Festlegung**
4000	**Abgeschlossen**
5000	**Verstetigung**
9999	**Sonstiges**

Attribute:

Name: sonstGebietArt

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte Klassifikation einer nicht auf dem BauGB beruhenden, z.B. länderspezifischen Gebietsausweisung. In dem Fall muss das Attribut "gebietsArt" den Wert 9999 (Sonstiges) haben.

Voidable: false

Multiplicity: 0..1

Value type: SO_SonstGebietArt (code list)

Attribute:

Name: sonstRechtsstandGebiet

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierter sonstiger Rechtsstand der Gebietsausweisung, der nicht durch die Liste SO_RechtsstandGebietTyp wiedergegeben werden kann. Das Attribut "rechtsstandGebiet" muss in diesem Fall den Wert 9999 (Sonstiges) haben.

Voidable: false

Multiplicity:	0..1
Value type:	SO_SonstRechtsstandGebietTyp (code list)
Attribute:	
Name:	traegerMassnahme
Definition:	From the XPlanung 5.2 specification: Maßnahmen-Träger.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.75 SO_Gewaesser

SO_Gewaesser	
Definition:	From the XPlanung 5.2 specification: Abbildung eines bestehenden Gewässers.
Subtype of:	ZoningElement SO_Objekt
Type:	Feature type
Attribute:	
Name:	artDerFestlegung
Definition:	From the XPlanung 5.2 specification: Klassifizierung des Gewässers.
Voidable:	false
Multiplicity:	0..1
Value type:	SO_KlassifizGewaesser (enumeration)
Values	1000 **Gewaesser** 10000 **Gewaesser1Ordnung** 10001 **Gewaesser2Ordnung** 10002 **Gewaesser3Ordnung** 10003 **StehendesGewaesser** 2000 **Hafen** 9999 **Sonstiges**
Attribute:	

Name:	detailArtDerFestlegung
Definition:	From the XPlanung 5.2 specification: Über eine Codeliste definierte detailliertere Klassifizierung des Gewässers.
Voidable:	false
Multiplicity:	0..1
Value type:	SO_DetailKlassifizGewaesser (code list)
Attribute:	
Name:	name
Definition:	From the XPlanung 5.2 specification: Informelle Bezeichnung dees Gewässers.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	nummer
Definition:	From the XPlanung 5.2 specification: Amtliche Bezeichnung / Kennziffer des Gewässers.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.76 SO_Objekt

SO_Objekt	
Definition:	From the XPlanung 5.2 specification:
Subtype of:	XP_Objekt
Supertype of:	SO_Denkmalschutzrecht SO_Gebiet SO_Gewaesser SO_SonstigesRecht
Type:	Feature type
Abstract:	true

1.5.77 SO_SonstigesRecht

SO_SonstigesRecht

Definition: From the XPlanung 5.2 specification:

Sonstige Festlegung.

Subtype of: ZoningElement

SO_Objekt

Type: Feature type

Attribute:

Name: artDerFestlegung

Definition: From the XPlanung 5.2 specification:

Rechtliche Klassifizierung der Festlegung.

Voidable: false

Multiplicity: 0..1

Value type: SO_KlassifizNachSonstigemRecht (enumeration)

Values

1000	**Bauschutzbereich**
1100	**Berggesetz**
1200	**Richtfunkverbindung**
1300	**Truppenuebungsplatz**
1400	**VermessungsKatasterrecht**
1500	**Rekultivierungsflaeche**
1600	**Renaturierungsflaeche**
9999	**Sonstiges**

Attribute:

Name: detailArtDerFestlegung

Definition: From the XPlanung 5.2 specification:

Über eine Codeliste definierte detailliertere rechtliche Klassifizierung der Festlegung.

Voidable: false

Multiplicity: 0..1

Value type: SO_DetailKlassifizNachSonstigemRecht (code list)

Attribute:

Name: nummer

Definition: From the XPlanung 5.2 specification:

	Amtliche Bezeichnung / Kennziffer der Festlegung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.78 XP_Hoehenangabe

XP_Hoehenangabe	
Definition:	From the XPlanung 5.2 specification: Spezifikation einer Angabe zur vertikalen Höhe oder zu einem Bereich vertikaler Höhen. Es ist möglich, spezifische Höhenangaben (z.B. die First- oder Traufhöhe eines Gebäudes) vorzugeben oder einzuschränken, oder den Gültigkeitsbereich eines Planinhalts auf eine bestimmte Höhe (hZwingend) bzw. einen Höhenbereich (hMin - hMax) zu beschränken, was vor allem bei der höhenabhängigen Festsetzung einer überbaubaren Grundstücksfläche (BP_UeberbaubareGrundstuecksflaeche), einer Baulinie (BP_Baulinie) oder einer Baugrenze (BP_Baugrenze) relevant ist. In diesem Fall bleiben die Attribute bezugspunkt und abweichenderBezugspunkt unbelegt.
Type:	Data type
Attribute:	
Name:	abweichenderBezugspunkt
Definition:	From the XPlanung 5.2 specification: Textuelle Spezifikation eines Höhenbezugspunktes wenn das Attribut "bezugspunkt" nicht belegt ist.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	abweichenderHoehenbezug
Definition:	From the XPlanung 5.2 specification: Textuelle Spezifikation des Höhenbezuges wenn das Attribut "hoehenbezug" nicht belegt ist.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name: bezugspunkt

Definition: From the XPlanung 5.2 specification:

Bestimmung des Bezugspunktes der Höhenangaben. Wenn weder dies Attribut noch das Attribut "abweichenderBezugspunkt" belegt sind, soll die Höhenangabe als vertikale Einschränkung des zugeordneten Planinhalts interpretiert werden.

Voidable: false

Multiplicity: 1

Value type: XP_ArtHoehenbezugspunkt (enumeration)

Values

1000	**TH**
2000	**FH**
3000	**OK**
3500	**LH**
4000	**SH**
4500	**EFH**
5000	**HBA**
5500	**UK**
6000	**GBH**
6500	**WH**

Attribute:

Name: h

Definition: From the XPlanung 5.2 specification:

Maximal zulässige Höhe des Bezugspunktes (bezugspunkt) .

Voidable: false

Multiplicity: 0..1

Value type: Length

Attribute:

Name: hMax

Definition: From the XPlanung 5.2 specification:

Maximal zulässige Höhe des Bezugspunktes (bezugspunkt) bei einer Bereichsangabe, bzw. obere Grenze des vertikalen Gültigkeitsbereiches eines Planinhalts, wenn "bezugspunkt"

	nicht belegt ist. In diesem Fall gilt: Ist "hMin" nicht belegt, gilt die Festlegung bis zur Höhe "hMax".
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	hMin
Definition:	From the XPlanung 5.2 specification: Minimal zulässige Höhe des Bezugspunktes (bezugspunkt) bei einer Bereichsangabe, bzw. untere Grenze des vertikalen Gültigkeitsbereiches eines Planinhalts, wenn "bezugspunkt" nicht belegt ist. In diesem Fall gilt: Ist "hMax" nicht belegt, gilt die Festlegung ab der Höhe "hMin".
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	hZwingend
Definition:	From the XPlanung 5.2 specification: Zwingend einzuhaltende Höhe des Bezugspunktes (bezugspunkt) , bzw. Beschränkung der vertikalen Gültigkeitsbereiches eines Planinhalts auf eine bestimmte Höhe.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	hoehenbezug
Definition:	From the XPlanung 5.2 specification: Art des Höhenbezuges.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_ArtHoehenbezug (enumeration)
Values	1000 **TH** 2000 **FH**

	3000	**OK**
	3500	**LH**
	4000	**SH**
	4500	**EFH**
	5000	**HBA**
	5500	**UK**
	6000	**GBH**
	6500	**WH**

1.5.79 XP_Objekt

XP_Objekt	
Definition:	From the XPlanung 5.2 specification: Abstrakte Oberklasse für alle XPlanung-Fachobjekte. Die Attribute dieser Klasse werden über den Vererbungs-Mechanismus an alle Fachobjekte weitergegeben.
Supertype of:	BP_Objekt FP_Objekt RP_Objekt SO_Objekt
Type:	Feature type
Abstract:	true
Attribute:	
Name:	aufschrift
Definition:	From the XPlanung 5.2 specification: Spezifischer Text zur Beschriftung von Planinhalten.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	endeBedingung
Definition:	From the XPlanung 5.2 specification: Notwendige Bedingung für das Ende der Wirksamkeit eines Planinhalts.
Voidable:	false

Multiplicity:	0..1
Value type:	XP_Wirksamkeitsbedingung (data type)
Attribute:	
Name:	gesetzlicheGrundlage
Definition:	From the XPlanung 5.2 specification: Angabe der gesetzlichen Grundlage des Planinhalts.
Voidable:	false
Multiplicity:	0..1
Value type:	XP_GesetzlicheGrundlage (code list)
Attribute:	
Name:	gliederung1
Definition:	From the XPlanung 5.2 specification: Kennung im Plan für eine erste Gliederungsebene (z.B. GE-E für ein "Eingeschränktes Gewerbegebiet").
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	gliederung2
Definition:	From the XPlanung 5.2 specification: Kennung im Plan für eine zweite Gliederungsebene (z.B. GE-E 3 für die "Variante 3 eines eingeschränkten Gewerbegebiets").
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	hoehenangabe
Definition:	From the XPlanung 5.2 specification: Angaben zur vertikalen Lage und Höhe eines Planinhalts.
Voidable:	false
Multiplicity:	0..*
Value type:	XP_Hoehenangabe (data type)

Attribute:

Name: rechtsstand

Definition: From the XPlanung 5.2 specification:

Angabe, ob der Planinhalt bereits besteht, geplant ist, oder zukünftig wegfallen soll.

Voidable: false

Multiplicity: 0..1

Value type: XP_Rechtsstand (enumeration)

Values

1000	**Geplant**
2000	**Bestehend**
3000	**Fortfallend**

Attribute:

Name: startBedingung

Definition: From the XPlanung 5.2 specification:

Notwendige Bedingung für die Wirksamkeit eines Planinhalts.

Voidable: false

Multiplicity: 0..1

Value type: XP_Wirksamkeitsbedingung (data type)

Attribute:

Name: text

Definition: From the XPlanung 5.2 specification:

Beliebiger Text.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

1.5.80 XP_SPEMassnahmenDaten

XP_SPEMassnahmenDaten

Definition: From the XPlanung 5.2 specification:

Spezifikation der Attribute für einer Schutz-, Pflege- oder Entwicklungsmaßnahme.

Type: Data type

Attribute:

Name: klassifizMassnahme

Definition: From the XPlanung 5.2 specification:

Klassifikation der Maßnahme.

Voidable: false

Multiplicity: 0..1

Value type: XP_SPEMassnahmenTypen (enumeration)

Values

1000	**ArtenreicherGehoelzbestand**
1100	**NaturnaherWald**
1200	**ExtensivesGruenland**
1300	**Feuchtgruenland**
1400	**Obstwiese**
1500	**NaturnaherUferbereich**
1600	**Roehrichtzone**
1700	**Ackerrandstreifen**
1800	**Ackerbrache**
1900	**Gruenlandbrache**
2000	**Sukzessionsflaeche**
2100	**Hochstaudenflur**
2200	**Trockenrasen**
2300	**Heide**
9999	**Sonstiges**

Attribute:

Name: massnahmeKuerzel

Definition: From the XPlanung 5.2 specification:

Kürzel der durchzuführenden Maßnahme.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: massnahmeText

Definition: From the XPlanung 5.2 specification:

Durchzuführende Maßnahme als freier Text.

Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.5.81 XP_Wirksamkeitsbedingung

XP_Wirksamkeitsbedingung	
Definition:	From the XPlanung 5.2 specification: Spezifikation von Bedingungen für die Wirksamkeit oder Unwirksamkeit einer Festsetzung.
Type:	Data type
Attribute:	
Name:	bedingung
Definition:	From the XPlanung 5.2 specification: Textlich formulierte Bedingung für die Wirksamkeit oder Unwirksamkeit einer Festsetzung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	datumAbsolut
Definition:	From the XPlanung 5.2 specification: Datum an dem eine Festsetzung wirksam oder unwirksam wird.
Voidable:	false
Multiplicity:	0..1
Value type:	Date

1.6 Package: LandAdministrationCadastre

Definition:

The LandAdministrationCadastre package provides classes on different levels, i.e. from international to national scope, to represent objects of the cadastre.

Sub-package:

Package: AAA

Package: LandAdministrationCadastre_INSPIRE

Package: LandAdministrationCadastre_LADM

Parent package:

Application schema: Siteplan

Diagram(s):

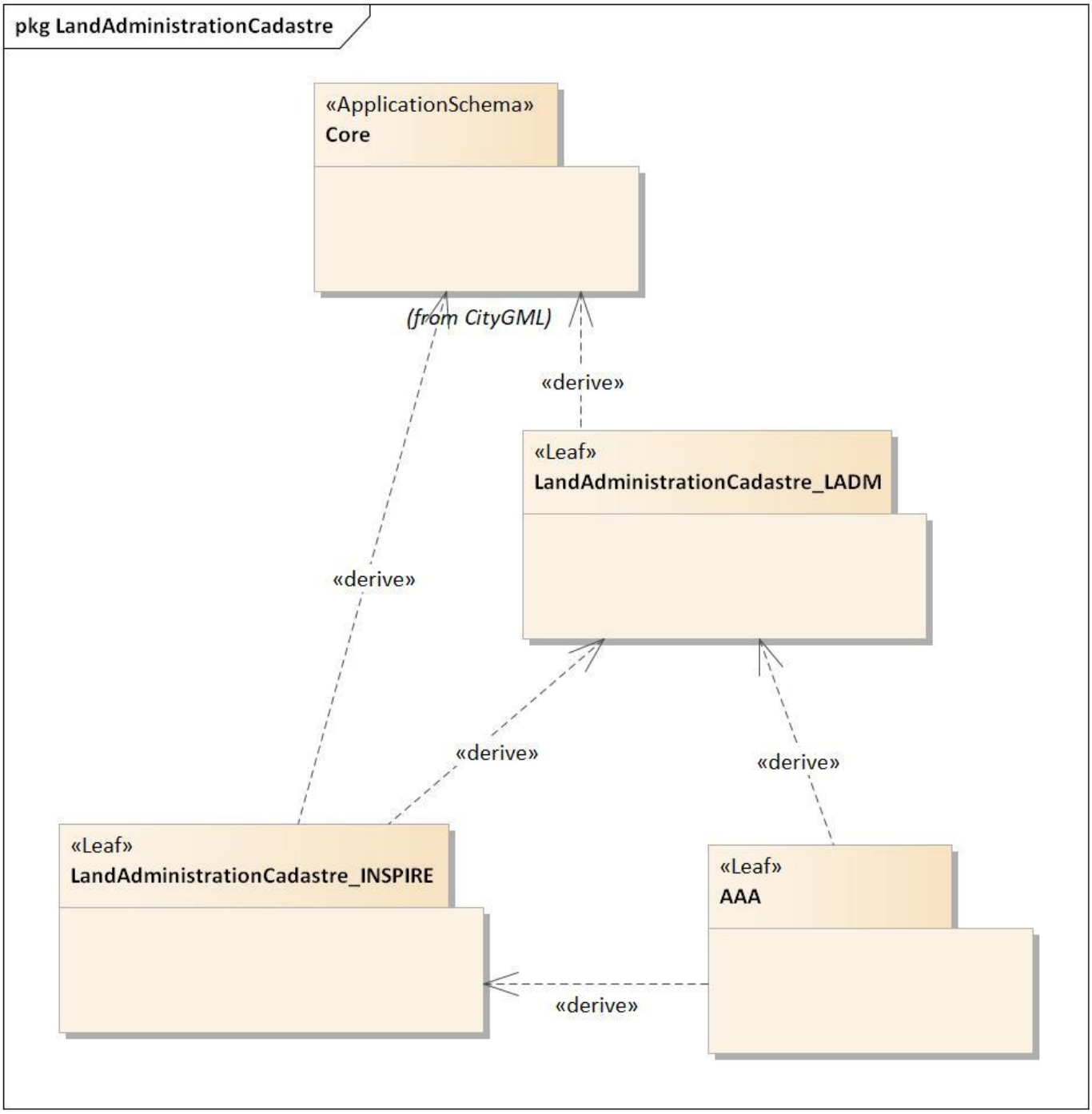


Diagram 21 - LandAdministrationCadastre

1.7 Package: LandAdministrationCadastre_LADM

Parent package:

Package: LandAdministrationCadastre

Diagram(s):

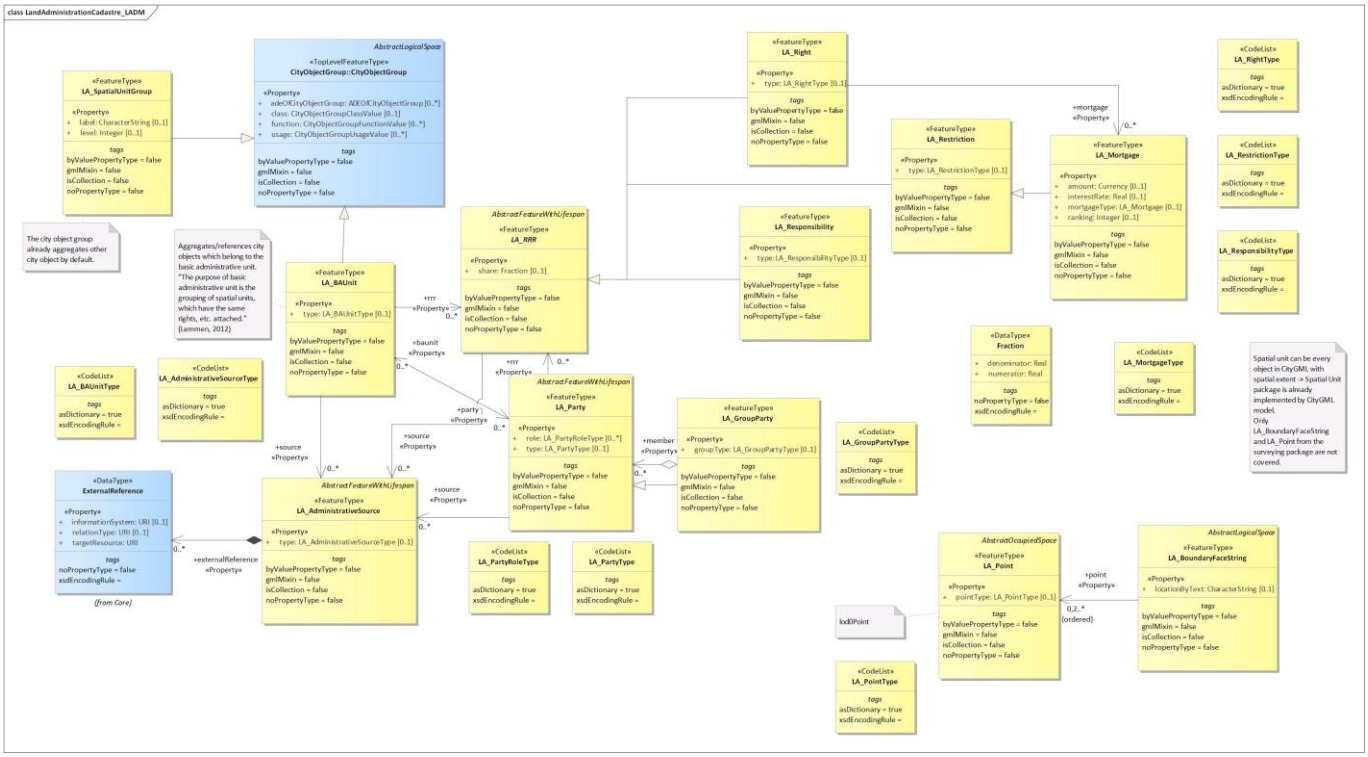


Diagram 22 - LandAdministrationCadastre_LADM

1.7.1 Fraction

Fraction	
Definition:	Describing a number as fraction.
Type:	Data type
Attribute:	
Name:	denominator
Definition:	Denominator.
Voidable:	false
Multiplicity:	1
Value type:	Real
Attribute:	
Name:	numerator
Definition:	Numerator.
Voidable:	false
Multiplicity:	1
Value type:	Real

1.7.2 LA_AdministrativeSource

LA_AdministrativeSource

Definition:	From the LADM specification: Source with the administrative description (where applicable) of the parties involved, the rights, restrictions and responsibilities created and the basic administrative units affected.
Subtype of:	AbstractFeatureWithLifespan
Supertype of:	AX_Buchungsblatt OfficialDocumentation
Type:	Feature type
Association role	
Name:	externalReference
Voidable:	false
Multiplicity:	0..*
Attribute:	
Name:	type
Definition:	From the LADM specification: Type of the document.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_AdministrativeSourceType (code list)

1.7.3 LA_BAUnit

LA_BAUnit	
Definition:	From the LADM specification: Basic administrative unit consisting of spatial units against which one ore more unique and homogenous rights, responsibilities or restrictions are associated to the whole entity as included in the land administration system.
Subtype of:	CityObjectGroup
Supertype of:	BasicPropertyUnit
Type:	Feature type
Association role	
Name:	party
Voidable:	false
Multiplicity:	0..*

Value type:	LA_Party (feature type)
Association role	
Name:	rrr
Voidable:	false
Multiplicity:	0..*
Value type:	LA_RRR (feature type)
Association role	
Name:	source
Voidable:	false
Multiplicity:	0..*
Value type:	LA_AdministrativeSource (feature type)
Attribute:	
Name:	type
Definition:	From the LADM specification: Type of the basic administrative unit.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_BAUnitType (code list)

1.7.4 LA_BoundaryFaceString

LA_BoundaryFaceString	
Definition:	From the LADM specification: Boundary forming part of the outside of a spatial unit.
Subtype of:	AbstractLogicalSpace
Supertype of:	CadastralBoundary
Type:	Feature type
Attribute:	
Name:	locationByText
Definition:	From the LADM specification: The boundary represented in text.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Association role	
Name:	point
Voidable:	false
Multiplicity:	0..*
Value type:	LA_Point (feature type)

1.7.5 LA_GroupParty

LA_GroupParty	
Definition:	From the LADM specification: Any number of parties, forming together a distinct entity, with each party registered.
Subtype of:	LA_Party
Supertype of:	AX_PersonenGruppe
Type:	Feature type
Attribute:	
Name:	groupType
Definition:	From the LADM specification: The type of the group party.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_GroupPartyType (code list)
Association role	
Name:	member
Voidable:	false
Multiplicity:	0..*
Value type:	LA_Party (feature type)

1.7.6 LA_Mortgage

LA_Mortgage	
Definition:	From the LADM specification: Lien on real property used to secure a debt.
Subtype of:	LA_Restriction
Type:	Feature type

Attribute:	
Name:	amount
Definition:	From the LADM specification: The amount of money of the mortgage.
Voidable:	false
Multiplicity:	0..1
Value type:	Currency
Attribute:	
Name:	interestRate
Definition:	From the LADM specification: Interest rate of the mortgage (percentage).
Voidable:	false
Multiplicity:	0..1
Value type:	Real
Attribute:	
Name:	mortgageType
Definition:	From the LADM specification: Type of the mortgage.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_Mortgage (feature type)
Attribute:	
Name:	ranking
Definition:	From the LADM specification: The ranking order if more than one mortgage applies to a right.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.7.7 LA_Party

LA_Party	
Definition:	From the LADM specification:

	A person or organisation that plays a role in a rights transaction.
Subtype of:	AbstractFeatureWithLifespan
Supertype of:	AX_Person
	AX_Vertretung
	LA_GroupParty
Type:	Feature type
Association role	
Name:	baunit
Voidable:	false
Multiplicity:	0..*
Value type:	LA_BAUnit (feature type)
Attribute:	
Name:	role
Definition:	From the LADM specification:
	The role of a party in the data update and maintenance process.
Voidable:	false
Multiplicity:	0..*
Value type:	LA_PartyRoleType (code list)
Association role	
Name:	rrr
Voidable:	false
Multiplicity:	0..*
Value type:	LA_RRR (feature type)
Association role	
Name:	source
Voidable:	false
Multiplicity:	0..*
Value type:	LA_AdministrativeSource (feature type)
Association role	
Name:	spunit
Voidable:	false

Multiplicity:	0..*
Value type:	SP_PlanningUnit (feature type)
Attribute:	
Name:	type
Definition:	From the LADM specification: The type of the party.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_PartyType (code list)

1.7.8 LA_Point

LA_Point	
Definition:	From the LADM specification: 0-dimensional geometric primitive, representing a position. Serving as superclass for the point classes from the AAA schema.
Subtype of:	AbstractOccupiedSpace
Supertype of:	AX_Festpunkt AX_Grenzpunkt AX_Netzpunkt
Type:	Feature type
Attribute:	
Name:	pointType
Definition:	From the LADM specification: The type of point.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_PointType (code list)

1.7.9 LA_RRR

LA_RRR	
Definition:	From the LADM specification: An instance of a subclass of LA_RRR is a right (or social tenure relationship), a restriction, or a responsibility.

Subtype of:	AbstractFeatureWithLifespan
Supertype of:	LA_Responsibility
	LA_Restriction
	LA_Right
Type:	Feature type
Abstract:	true
Attribute:	
Name:	share
Definition:	From the LADM specification:
Voidable:	false
Multiplicity:	0..1
Value type:	Fraction (data type)
Association role	
Name:	source
Voidable:	false
Multiplicity:	0..*
Value type:	LA_AdministrativeSource (feature type)

1.7.10 LA_Responsibility

LA_Responsibility	
Definition:	From the LADM specification:
	Formal or informal obligation to do something.
Subtype of:	LA_RRR
Type:	Feature type
Attribute:	
Name:	type
Definition:	From the LADM specification:
	Type of the responsibility.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_ResponsibilityType (code list)

1.7.11 LA_Restriction

LA_Restriction	
Definition:	From the LADM specification: Formal or informal entitlement to refrain from doing something.
Subtype of:	LA_RRR
Supertype of:	LA_Mortgage
Type:	Feature type
Attribute:	
Name:	type
Definition:	From the LADM specification: Type of restriction.
Voidable:	false
Multiplicity:	0..1
Value type:	LA_RestrictionType (code list)

1.7.12 LA_Right

LA_Right	
Definition:	From the LADM specification: Action, activity or class of actions that a system participant may perform on or using an associated resource.
Subtype of:	LA_RRR
Type:	Feature type
Association role	
Name:	mortgage
Voidable:	false
Multiplicity:	0..*
Value type:	LA_Mortgage (feature type)
Attribute:	
Name:	type
Definition:	From the LADM specification: Type of the right.
Voidable:	false
Multiplicity:	0..1

Value type:	LA_RightType (code list)

1.7.13 LA_SpatialUnitGroup

LA_SpatialUnitGroup	
Definition:	From the LADM specifiaction: Any number of spatial units, considered as an entity.
Subtype of:	CityObjectGroup
Supertype of:	CadastralZoning
Type:	Feature type
Attribute:	
Name:	label
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	level
Definition:	From the LADM specification: The level in the hierarchy of the (administrative or zoning) subdivision.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.8 Package: LandAdministrationCadastre_INSPIRE

Parent package:

Package: LandAdministrationCadastre

Diagram(s):

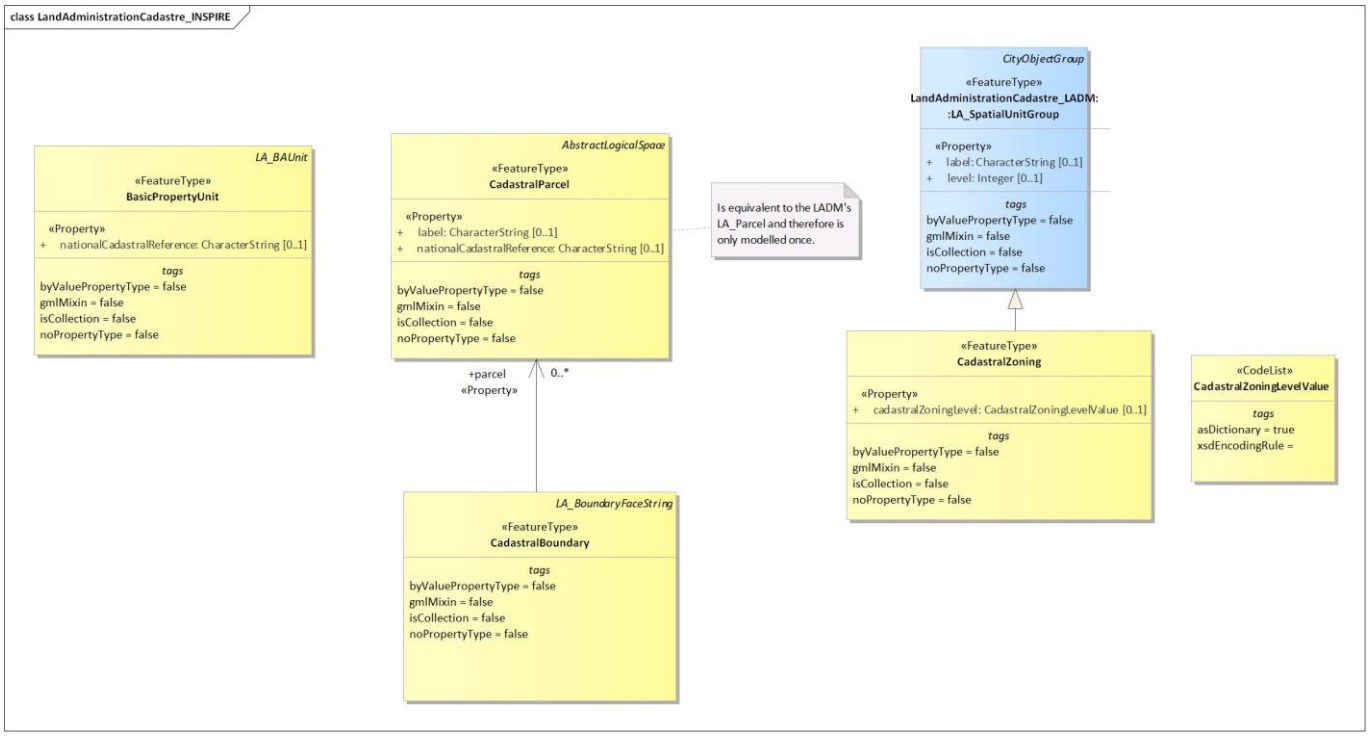


Diagram 23 - LandAdministrationCadastre_INSPIRE

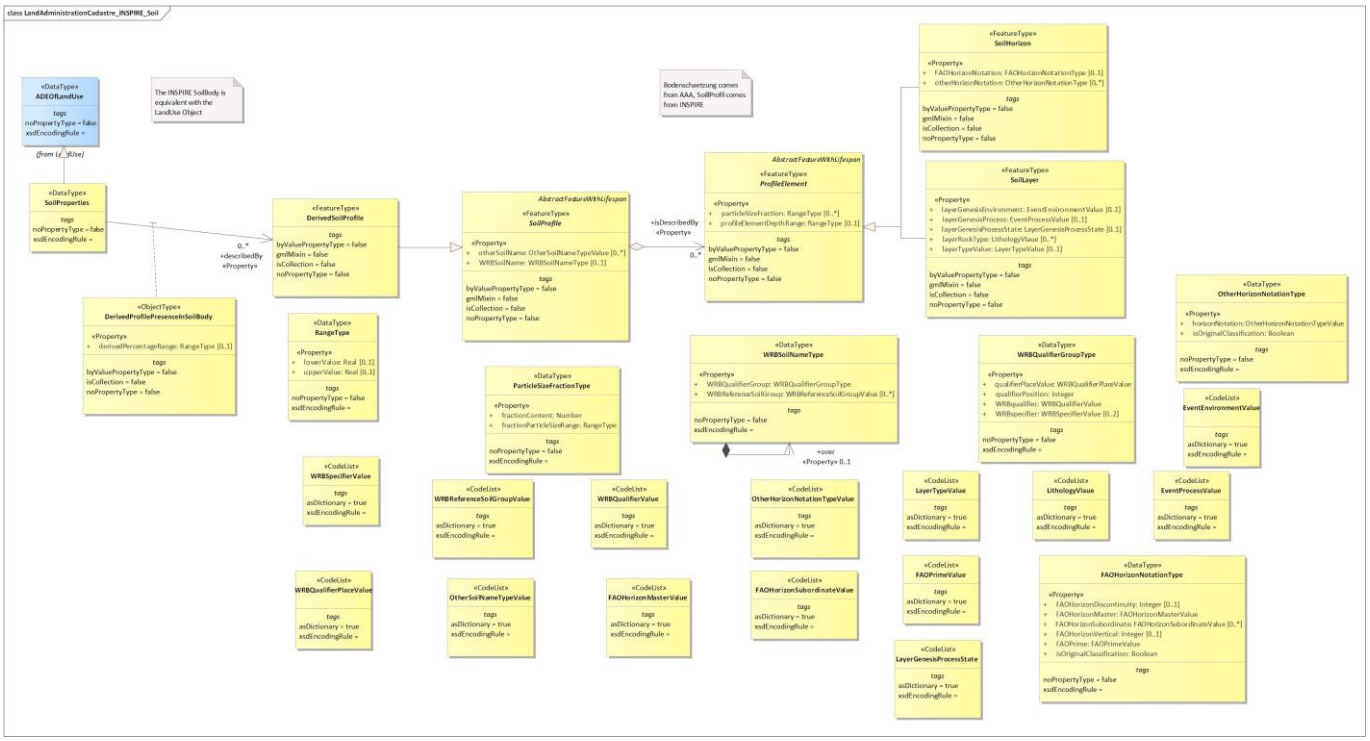


Diagram 24 - LandAdministrationCadastre_INSPIRE_Soil

1.8.1 BasicPropertyUnit

BasicPropertyUnit	
Definition:	From the INSPIRE specification: The basic unit of ownership that is recorded in the land books, land registers or equivalent. It is defined by unique ownership and homogeneous real property rights, and may

	consist of one or more adjacent or geographically separate parcels. SOURCE Adapted from UN ECE 2004.
Subtype of:	LA_BAUnit
Supertype of:	AX_Buchungstelle
Type:	Feature type
Attribute:	
Name:	nationalCadastralReference
Definition:	From the INSPIRE specification: Thematic identifier at national level, generally the full national code of the basic property unit. Must ensure the link to the national cadastral register or equivalent. The national cadastral reference can be used also in further queries in national services.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.8.2 CadastralBoundary

CadastralBoundary	
Definition:	From the INSPIRE specification: Part of the outline of a cadastral parcel. One cadastral boundary may be shared by two neighbouring cadastral parcels. In the INSPIRE context, cadastral boundaries are to be made available by member states where absolute positional accuracy information is recorded for the cadastral boundary (attribute estimated accuracy)
Subtype of:	LA_BoundaryFaceString
Supertype of:	AX_BesondereFlurstuecksgrenze
Type:	Feature type
Association role	
Name:	parcel
Voidable:	false
Multiplicity:	0..*
Value type:	CadastralParcel (feature type)

1.8.3 CadastralParcel

CadastralParcel	
Definition:	From the INSPIRE specification: Areas defined by cadastral registers or equivalent. SOURCE [INSPIRE Directive:2007]. As much as possible, in the INSPIRE context, cadastral parcels should be forming a partition of national territory. Cadastral parcel should be considered as a single area of Earth surface (land and/or water), under homogeneous real property rights and unique ownership, real property rights and ownership being defined by national law (adapted from UN ECE 2004 and WG-CPI, 2006). By unique ownership is meant that the ownership is held by one or several joint owners for the whole parcel.
Subtype of:	AbstractLogicalSpace
Supertype of:	AX_Flurstueck
Type:	Feature type
Attribute:	
Name:	label
Definition:	From the INSPIRE specification: Text commonly used to display the cadastral parcel identification.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	nationalCadastralReference
Definition:	From the INSPIRE specification: Thematic identifier at national level, generally the full national code of the cadastral parcel. Must ensure the link to the national cadastral register or equivalent.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.8.4 CadastralZoning

CadastralZoning	
Definition:	From the INSPIRE specification: Intermediary areas used in order to divide national territory into cadastral parcels. In the INSPIRE context, cadastral zonings are to be used to carry metadata information and to facilitate portrayal and search of data. Cadastral zonings have generally been defined when cadastral maps were created for the first time
Subtype of:	LA_SpatialUnitGroup
Supertype of:	AX_Gemarkung
Type:	Feature type
Attribute:	
Name:	cadastralZoningLevel
Definition:	From the INSPIRE specification: Level of the cadastral zoning in the national cadastral hierarchy.
Voidable:	false
Multiplicity:	0..1
Value type:	CadastralZoningLevelValue (code list)

1.8.5 DerivedProfilePresenceInSoilBody

DerivedProfilePresenceInSoilBody	
Definition:	From the INSPIRE specification: Indicates the percentages (lower and upper boundary) that the derived profile takes part in the Soil body. NOTE 1 A soil body is characterized by one or more derived soil profiles in a defined geographic area. When the soil body is characterized by more than one derived profiles, the distribution area of these derived soil profiles is not spatially defined, but their presence is indicated by a range of percentages. NOTE 2 The sum of lower boundary parts should not exceed 100%. NOTE 3 If not a range, but a specific percentage is used then the lower and upper boundaries are equal.
Type:	Object type

Attribute:	
Name:	derivedPercentageRange
Definition:	From the INSPIRE specification: Interval that defines the minimum and maximum percentage of the area of the soil body represented by a specific derived soil profile.
Voidable:	false
Multiplicity:	0..1
Value type:	RangeType (data type)

1.8.6 DerivedSoilProfile

DerivedSoilProfile	
Definition:	From the INSPIRE specification: Non-point-located soil profile that serves as a reference profile for a specific soil type in a certain geographical area. The characteristics of a derived soil profile are mostly derived (e.g. averaged) from one or several observed profiles of the same soil type in the area of interest, or are designed with expert knowledge about the same kind of landscape. NOTE 1 The derived soil profile represents the average or typical profile that characterizes the so called soil typological unit, soil series.
Subtype of:	SoilProfile
Supertype of:	AX_Bodenschaetzung
Type:	Feature type

1.8.7 FAOHorizonNotationType

FAOHorizonNotationType	
Definition:	From the INSPIRE specification: A classification of a horizon according to the Horizon classification system specified in Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006. A code system that denotes horizons in which the same dominant soil forming processes has been active similarly. This code summarizes many observations of the soil description and gives an impression about the genetic processes that have formed the soil under observation.

	NOTE The horizon notation according to FAO (2006) is a combination of several symbols: - A number that gives information about discontinuities, i.e. the number of the material in which the soil has formed, counted up from the soil surface, but not for the first material. - One or two capital letters that designate the type of master horizon (or transitional horizon), possibly separated by a slash - Lower case letters that designate subordinate characteristics of the horizon. - A number that designates horizontal subdivisions of otherwise similarly denoted horizon parts - A prime that enables to distinguish two horizons that have the same naming, but formed in different cycles of pedogenesis. EXAMPLE 2B' tg1 SOURCE 1 Page 67 - 77 of the Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006. (ISBN 92-5-105521-1)
Type:	Data type
Attribute:	
Name:	FAOHorizonDiscontinuity
Definition:	From the INSPIRE specification: number used to indicate a discontinuity in the horizon notation. In mineral soils, Arabic numerals are used as prefixes to indicate discontinuities. Wherever needed, they are used preceding A, E, B, C and R. They are not used with I and W, although these symbols clearly indicate a discontinuity. These prefixes are distinct from Arabic numerals used as suffixes to denote vertical subdivisions. A discontinuity is a significant change in particle-size distribution or mineralogy that indicates a difference in the material from which the horizons formed or a significant difference in age or both, unless that difference in age is indicated by the suffix b. Symbols to identify discontinuities are used only when they will contribute substantially to the reader’s understanding of relationships among horizons. The stratification common in soils formed in alluvium is not designated as discontinuities unless particle-size distribution differs markedly from layer to layer even though genetic horizons have formed in the contrasting layers.

	Where a soil has formed entirely in one kind of material, a prefix is omitted from the symbol; the whole profile is material 1. Similarly, the uppermost material in a profile having two or more contrasting materials is understood to be material 1, but the number is omitted. Numbering starts with the second layer of contrasting material, which is designated 2. Underlying contrasting layers are numbered consecutively. SOURCE: Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006. NOTE a discontinuity is a boundary between two geogenic layers.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	FAOHorizonMaster
Definition:	From the INSPIRE specification: Symbol of the master part of the horizon notation. SOURCE Guidelines for soil description (4th ed.) FAO 2006 p. 67
Voidable:	false
Multiplicity:	1
Value type:	FAOHorizonMasterValue (code list)
Attribute:	
Name:	FAOHorizonSubordinate
Definition:	From the INSPIRE specification: Designations of subordinate distinctions and features within the master horizons and layers are based on profile characteristics observable in the field and are applied during the description of the soil at the site. Lower case letters are used as suffixes to designate specific kinds of master horizons and layers, and other features. SOURCE Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006. table 85
Voidable:	false
Multiplicity:	0..*

Value type:	FAOHorizonSubordinateValue (code list)
Attribute:	
Name:	FAOHorizonVertical
Definition:	From the INSPIRE specification: Order number of the vertical subdivision in the horizon notation. The number is used to designate the horizontal subdivision of a horizon identified by a single set of letter symbol on the basis of structure, texture, colour, etc. The number 1 is used to designate the upper part of the horizon. The number 2 the part of the horizon situated below, etc. If there is no horizontal subdivision made, no number is given. NOTE the use of the word vertical is misleading in the Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006., because the horizons are separated in two or several horizontal subdomains. Vertical denotes the order in vertical direction.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	FAOPrime
Definition:	From the INSPIRE specification: A prime and double prime may be used to connotate master horizon symbol of the lower of two respectively three horizons having identical Arabic-numeral prefixes and letter combinations. Identical designations may be appropriate for two or more horizons or layers separated by at least one horizon or layer of a different kind in the same pedon. The sequence A-E-Bt-E-Btx-C is an example - the soil has two E horizons. To make communication easier, a prime is used with the master horizon symbol of the lower of two horizons having identical letter designations: A-E-Bt-E'-Btx-C.The prime is applied to the capital letter designation, and any lower case symbol follows it: B't. The prime is not used unless all letters of the designations of two different layers are identical. Prime can be used for both minerals or organic soils.

	SOURCE: Guidelines for soil description, 4th edition, Food and Agriculture Organization of the United Nations, Rome, 2006.
Voidable:	false
Multiplicity:	1
Value type:	FAOPrimeValue (code list)
Attribute:	
Name:	isOriginalClassification
Definition:	From the INSPIRE specification: Boolean value to indicate whether the FAO horizon notation was the original notation to describe the horizon. This is to indicate whether the horizon notation has been determined in the field (or for derived profiles during the original derivation), or has been adapted, changed or assigned on the basis of the original horizon description later. This might be relevant data quality information.
Voidable:	false
Multiplicity:	1
Value type:	Boolean

1.8.8 OtherHorizonNotationType

OtherHorizonNotationType	
Definition:	From the INSPIRE specification:
Type:	Data type
Attribute:	
Name:	horizonNotation
Definition:	From the INSPIRE specification:
Voidable:	false
Multiplicity:	1
Value type:	OtherHorizonNotationTypeValue (code list)
Attribute:	
Name:	isOriginalClassification
Definition:	From the INSPIRE specification:
Voidable:	false
Multiplicity:	1

Value type:	Boolean

1.8.9 ParticleSizeFractionType

ParticleSizeFractionType	
Definition:	From the INSPIRE specification: Share of the soil that is composed of mineral soil particles of the size within the size range specified. Mineral part of the soil, fractioned on the basis of size (diameter), limits of the particles. It is the fine earth fraction. That is, the portion of the soil that passes through a 2 mm diameter sieve opening. The grain (or particle) size distribution characterizes the soil mineral material, based on the share of each equivalent diameter class of the individual particles. SOURCE1 NRCS Natural Resources Conservation Service, Soil survey laboratory information manual, Soil survey investigation report n.45 version 1.0 May 1995 pag. 11 SOURCE2 GLOSSARY OF SOIL SCIENCE TERMS
Type:	Data type
Attribute:	
Name:	fractionContent
Definition:	From the INSPIRE specification:
Voidable:	false
Multiplicity:	1
Value type:	Number
Attribute:	
Name:	fractionParticleSizeRange
Definition:	From the INSPIRE specification:
Voidable:	false
Multiplicity:	1
Value type:	RangeType (data type)

1.8.10 ProfileElement

ProfileElement	
Definition:	From the INSPIRE specification:

	An abstract spatial object type grouping soil layers and / or horizons for functional/operational aims. Profile element is the general term for both horizons and layers.
Subtype of:	AbstractFeatureWithLifespan
Supertype of:	SoilHorizon SoilLayer
Type:	Feature type
Abstract:	true
Attribute:	
Name:	particleSizeFraction
Definition:	From the INSPIRE specification: Particle size fraction. Mineral part of the soil, fractioned on the basis of size (diameter), limits of the particles. It indicates how much of the mineral soil material is composed of soil particles of the specified size range. SOURCE NRCS Natural Resources Conservation Service, Soil survey laboratory information manual, Soil survey investigation report n.45 version 1.0 May 1995 pag. 11
Voidable:	false
Multiplicity:	0..*
Value type:	RangeType (data type)
Attribute:	
Name:	profileElementDepthRange
Definition:	From the INSPIRE specification: Upper and lower depth of the profile element (layer or horizon) measured from the surface (0 cm) of a soil profile (in cm). Depth range consists of the average upper and lower depth of appearance of the profile element from the surface. NOTE Most soil boundaries are zones of transition rather than sharp lines of division. The average depth of the upper boundaries and the average depth of the lower boundaries of each profile element are given in centimetres, measured from the surface (including organic and mineral covers) of the soil downwards, i.e. all depth values are positive numbers.

	EXAMPLE H horizon 0-5 cm, A horizon 5-30 cm, B horizon 30-80 cm NOTE Following rules should be taken into account - lowerValue and upperValue: should be positive values, - upperValue is the depth from the top of the element (e.g. 20) - lowerValue is the depth of the bottom of the element (e.g. 40) - if only upperValue is indicated: it is assumed that the lowerValue is unknown; this is only possible for the deepest layer or horizon of a profile. - if only lowerValue is indicated: it is assumed that the upperValue equals 0, and thus the range is between 0 and the lowerValue value
Voidable:	false
Multiplicity:	0..1
Value type:	RangeType (data type)

1.8.11 RangeType

RangeType	
Definition:	From the INSPIRE specification: A range value defined by an upper limit and a lower limit.
Type:	Data type
Attribute:	
Name:	lowerValue
Definition:	From the INSPIRE specification: Value defining the lower limit of a specific property.
Voidable:	false
Multiplicity:	0..1
Value type:	Real
Attribute:	

Name:	upperValue
Definition:	From the INSPIRE specification: Value defining the upper limit of a specific property.
Voidable:	false
Multiplicity:	0..1
Value type:	Real

1.8.12 SoilHorizon

SoilHorizon	
Definition:	From the INSPIRE specification: Domain of a soil with a certain vertical extension, more or less parallel to the surface and homogeneous for most morphological and analytical characteristics, developed in a parent material layer through pedogenic processes or made up of in-situ sedimented organic residues of up-growing plants (peat). SOURCE ISO/WD 28258, modified NOTE Horizons may be part of a layer.
Subtype of:	ProfileElement
Type:	Feature type
Attribute:	
Name:	FAOHorizonNotation
Definition:	From the INSPIRE specification: Designation of the soil horizon. The FAO horizon notation is a code system characterizing horizons regarding the dominant soil forming processes that have been active during the horizon formation. This code summarizes many observations of the soil description and gives an impression about the genetic processes that have formed the soil under observation. NOTE The horizon notation is a combination of several letter, number and symbols. SOURCE Guidelines for soil description, 4th edition, Food and Agricultural Organization of the United Nations, Rome, 2006 EXAMPLE Bw: meaning B horizon with a development of colour and/or structure.
Voidable:	false

Multiplicity:	0..1
Value type:	FAOHorizonNotationType (data type)
Attribute:	
Name:	otherHorizonNotation
Definition:	From the INSPIRE specification: Designation of the soil horizon according to a specific classification system. A code system characterizing horizons regarding the dominant soil forming processes that have been active during the horizon formation. This code summarizes many observations of the soil description and gives information about the genetic processes that have formed the soil under observation.
Voidable:	false
Multiplicity:	0..*
Value type:	OtherHorizonNotationType (data type)

1.8.13 SoilLayer

SoilLayer	
Definition:	From the INSPIRE specification: Domain of a soil with a certain vertical extension developed through non-pedogenic processes, displaying a change in structure and/or composition to possibly over- or underlying adjacent domains, or a grouping of soil horizons or other sub-domains with a special purpose. NOTE1 Different kinds of layer concepts are covered by this definition. EXAMPLE 1 Geogenic layers: These are domains, resulting from e.g. sedimentation (as non-pedogenic) processes, that display an unconformity to possibly over- or underlying adjacent domains. EXAMPLE 2 Topsoil and Subsoil: These can be domains that group different soil horizon types (e.g. A vs. B horizons), or a special case of fixed depths with only two depth ranges (e.g. 0-15 cm: topsoil and, 15-75 cm: subsoil). EXAMPLE 3 Depth intervals: They are often used in soil monitoring, sampling of contaminated sites and in modelling and include: (i) depth increments (also called fixed depths) that are often used for sampling, e.g. 0-30cm, 30-60cm, and so on, (ii) a single depth range in which a soil sample (”specimen”) is taken and for which the analytical result is valid, and (iii) soil slicing, that is, profile segmentation

	according to a specified vector, for instance, either regularly spaced intervals (1cm), or a user-defined vector of segment boundaries (i.e. 0-10, 10-25, 25-50, 50-100). Slicing is used in modelling to generate continuous depth functions for soil properties. EXAMPLE 4: In the framework of soils deeply modified by human activity, artificial layers may be due to different kinds of deposits (concrete, bricks, …). SOURCE WD ISO28258, modified
Subtype of:	ProfileElement
Type:	Feature type
Attribute:	
Name:	layerGenesisEnvironment
Definition:	From the INSPIRE specification: Setting in which the last non-pedogenic process (geologic or anthropogenic) that coined the material composition and internal structure of the layer took place. The material in which a soil develops is influenced by the environmental situation in which the processes of its formation took place, e.g. sedimentation from water results in quite differently structured layers when it has been deposited in running water than in a lake.
Voidable:	false
Multiplicity:	0..1
Value type:	EventEnvironmentValue (code list)
Attribute:	
Name:	layerGenesisProcess
Definition:	From the INSPIRE specification: Last non-pedogenic process (geologic or anthropogenic) that coined the material composition and internal structure of the layer.
Voidable:	false
Multiplicity:	0..1
Value type:	EventProcessValue (code list)
Attribute:	
Name:	layerGenesisProcessState
Definition:	From the INSPIRE specification:

	Indication whether the process specified in layerGenesisProcess is on-going or seized in the past. Process state gives an idea whether current non-pedogenic processes affect the soil or not. E.g. on current floodplains, input of sediments during seasonal flooding events is received, with comparatively young soil development in it, while in older fluvial sediments that are no longer under a regime of seasonal or irregular flooding, soil development might be more advanced.
Voidable:	false
Multiplicity:	0..1
Value type:	LayerGenesisProcessState (code list)
Attribute:	
Name:	layerRockType
Definition:	From the INSPIRE specification: Type of the material in which the layer developed. Simplified list of terms to “classify” geologic units.
Voidable:	false
Multiplicity:	0..*
Value type:	LithologyVlaue (code list)
Attribute:	
Name:	layerTypeValue
Definition:	From the INSPIRE specification: Assignation of a layer according to the concept that fits its kind. EXAMPLE Topsoil: meaning the upper part of the natural mineral soil that is generally dark coloured and has a higher content of organic matter and nutrients when compared to the mineral horizons below.
Voidable:	false
Multiplicity:	0..1
Value type:	LayerTypeValue (code list)

1.8.14 SoilProfile

SoilProfile	
Definition:	From the INSPIRE specification:

	Description of the soil that is characterized by a vertical succession of profile elements. NOTE The soil profile is abstracted from observations in a trial pit or a boring, or derived from expert knowledge using other soil profiles.
Subtype of:	AbstractFeatureWithLifespan
Supertype of:	DerivedSoilProfile
Type:	Feature type
Abstract:	true
Attribute:	
Name:	WRBSoilName
Definition:	From the INSPIRE specification: Identification of the soil profile. NOTE The structure of the WRBSoilNameType was based on the World reference base for soil resources 2006, first update 2007. World Soil Resources Reports no 103. Food and Agriculture Organization of the United Nations, Rome 2007. EXAMPLE Lixic Vertic Ferralsol (Ferric, Rhodic); WRB version 2006 (update 2007) or 2010.
Voidable:	false
Multiplicity:	0..1
Value type:	WRBSoilNameType (data type)
Association role	
Name:	isDescribedBy
Voidable:	false
Multiplicity:	0..*
Value type:	ProfileElement (feature type)
Attribute:	
Name:	otherSoilName
Definition:	From the INSPIRE specification: Identification of the soil profile according to a specific classification scheme.
Voidable:	false
Multiplicity:	0..*
Value type:	OtherSoilNameTypeValue (code list)

1.8.15 SoilProperties

SoilProperties	
Definition:	Additional properties describing the soil composition of a piece of land.
Subtype of:	ADEOfLandUse
Type:	Data type
Association role	
Name:	describedBy
Voidable:	false
Multiplicity:	0..*
Value type:	DerivedSoilProfile (feature type)

1.8.16 WRBQualifierGroupType

WRBQualifierGroupType	
Definition:	From the INSPIRE specification: A data type to define the group of a qualifier and its possible specifier(s), its place and position with regard to the World Reference Base (WRB) Reference Soil Group (RSG) it belongs to according to World reference base for soil resources 2006, first update 2007, World Soil Resources Reports No. 103, Food and Agriculture Organization of the United Nations, Rome, 2007.
Type:	Data type
Attribute:	
Name:	WRBqualifier
Definition:	From the INSPIRE specification: Name element of WRB, 2nd level of classification. SOURCE World reference base for soil resources 2006, first update 2007, World Soil Resources Reports No. 103, Food and Agriculture Organization of the United Nations, Rome, 2007.
Voidable:	false
Multiplicity:	1
Value type:	WRBQualifierValue (code list)
Attribute:	
Name:	WRBspecifier

Definition:	From the INSPIRE specification: Code that indicates the degree of expression of a qualifier or the depth range of which the qualifier applies. Specifiers may be used to indicate depth of occurrence, or to express the intensity of soil characteristics. EXAMPLE Buried layers can be indicated by the specifier “Thapto”. NOTE 1 The specifier code is always added after the qualifier code. Exceptions are defined in the qualifier list of WRB. NOTE 2 Some specifiers can be combined with each other for one qualifier. SOURCE World reference base for soil resources 2006, first update 2007, World Soil Resources Reports No. 103, Food and Agriculture Organization of the United Nations, Rome, 2007.
Voidable:	false
Multiplicity:	0..2
Value type:	WRBSpecifierValue (code list)
Attribute:	
Name:	qualifierPlaceValue
Definition:	From the INSPIRE specification: Attribute to indicate the placement of the Qualifier with regard to the WRB reference soil group (RSG). The placement can be in front of the RSG i.e. 'prefix' or it can be behind the RSG i.e. 'suffix'.
Voidable:	false
Multiplicity:	1
Value type:	WRBQualifierPlaceValue (code list)
Attribute:	
Name:	qualifierPosition
Definition:	From the INSPIRE specification: Number to indicate the position of a qualifier with regard to the WRB reference soil group (RSG) it belongs to and with regard to its placement to that (RSG) i.e. as a prefix or a suffix. If there are one or more prefix qualifiers: one of the qualifiers is in position 1, the other qualifiers are in position 2, 3, etc.;

	position 1 is the position closest to the RSG; position 2 is the position second closest to the RSG; etc If there are one or more suffix qualifiers: one of the qualifiers is in position 1, the other qualifiers are in position 2, 3, etc.; position 1 is the position closest to the RSG; position 2 is the position second closest to the RSG; etc
Voidable:	false
Multiplicity:	1
Value type:	Integer

1.8.17 WRBSoilNameType

WRBSoilNameType	
Definition:	From the INSPIRE specification: An identification of the soil profile according to the profile to according to “World Reference Base for Soil Resources 2006, first update 2007”, World Soil Resources Reports No. 103. FAO, Rome. NOTE The structure of the WRBSoilNameType was based on the World reference base for soil resources 2006, first update 2007, World Soil Resources Reports No. 103, Food and Agriculture Organization of the United Nations, Rome, 2007. EXAMPLE Lixic Vertic Ferralsol (Ferric, Rhodic), WRB 2006, update 2007.
Type:	Data type
Attribute:	
Name:	WRBQualifierGroup
Definition:	From the INSPIRE specification: Data type to define the a group of a qualifier and its possible specifier(s), its place and position with regard to the WRBReferenceSoilGroup it belongs to.
Voidable:	false
Multiplicity:	1
Value type:	WRBQualifierGroupType (data type)
Attribute:	
Name:	WRBReferenceSoilGroup
Definition:	From the INSPIRE specification:

	First level of classification of the World Reference Base for Soil Resources. Reference Soil Groups are distinguished by the presence (or absence) of specific diagnostic horizons, properties and/or materials. NOTE The WRB soil classification system comprises 32 different RSGs. SOURCE World reference base for soil resources 2006, first update 2007, World Soil Resources Reports No. 103, Food and Agriculture Organization of the United Nations, Rome, 2007.
Voidable:	false
Multiplicity:	0..*
Value type:	WRBReferenceSoilGroupValue (code list)
Association role	
Name:	over
Voidable:	false
Multiplicity:	0..1

1.9 Package: AAA

Parent package:

Package: LandAdministrationCadastre

Diagram(s):

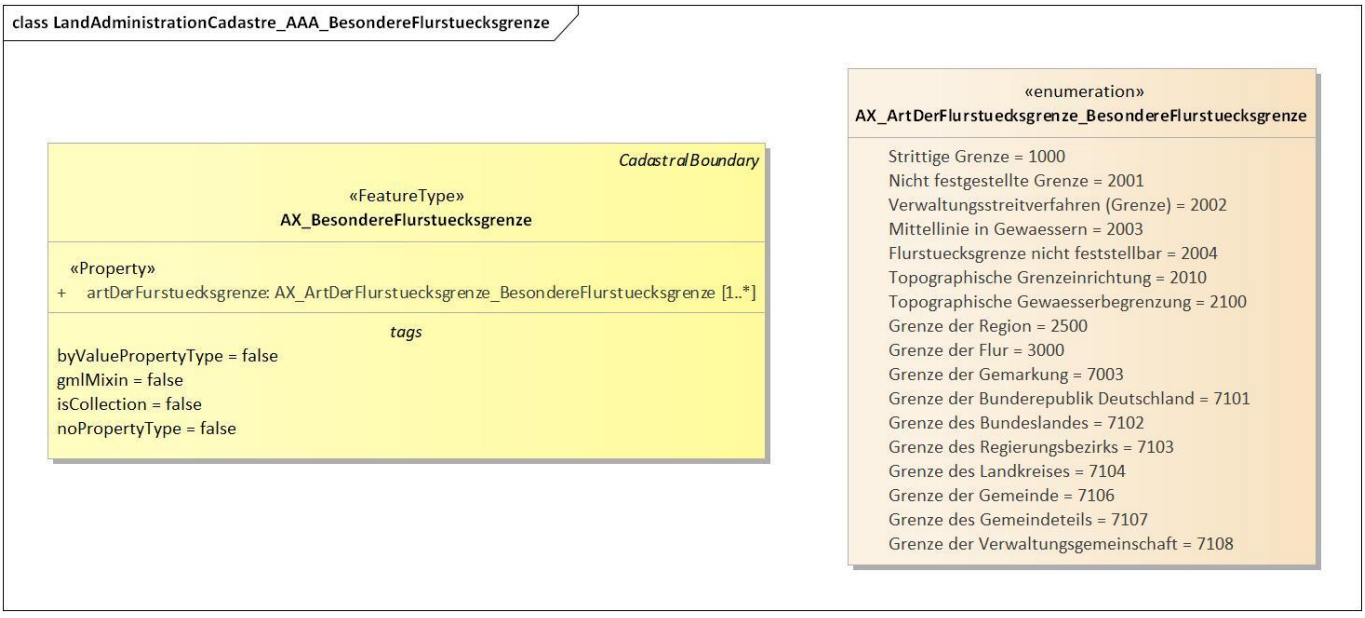


Diagram 25 - LandAdministrationCadastre_AAA_BesondereFlurstuecksgrenze

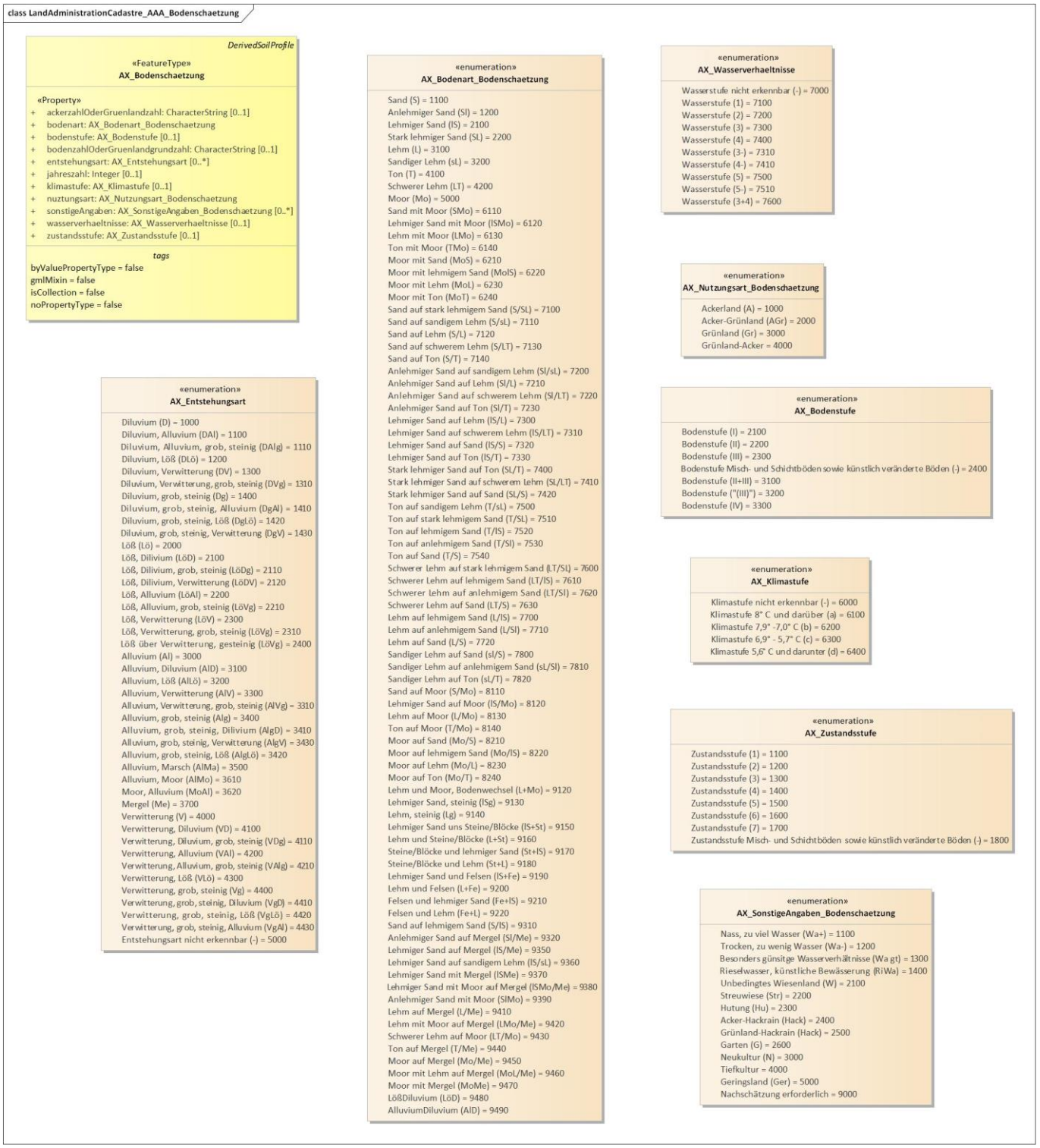


Diagram 26 - LandAdministrationCadastre_AAA_Bodenschaetzung

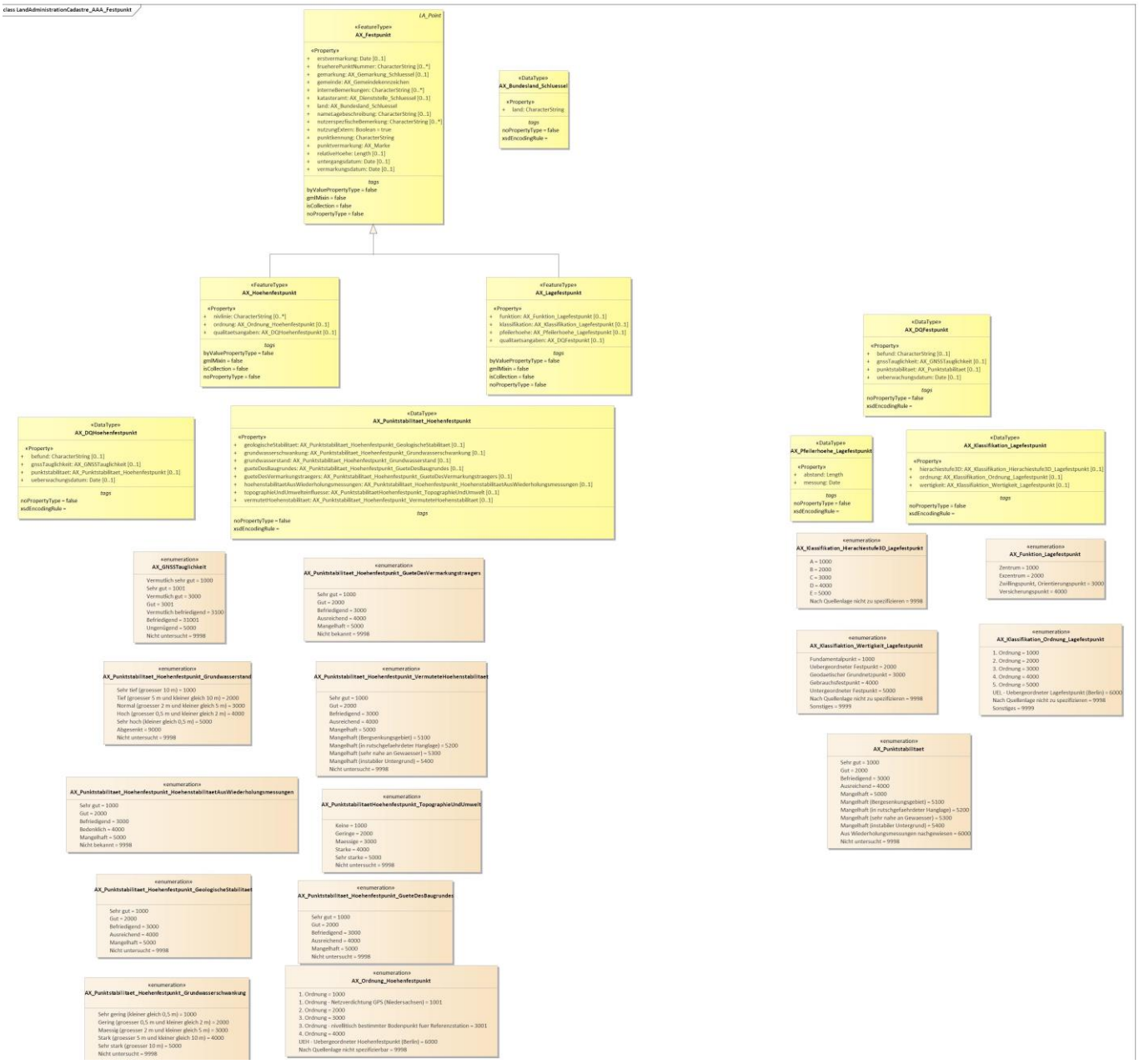


Diagram 27 - LandAdministrationCadastre_AAA_Festpunkt

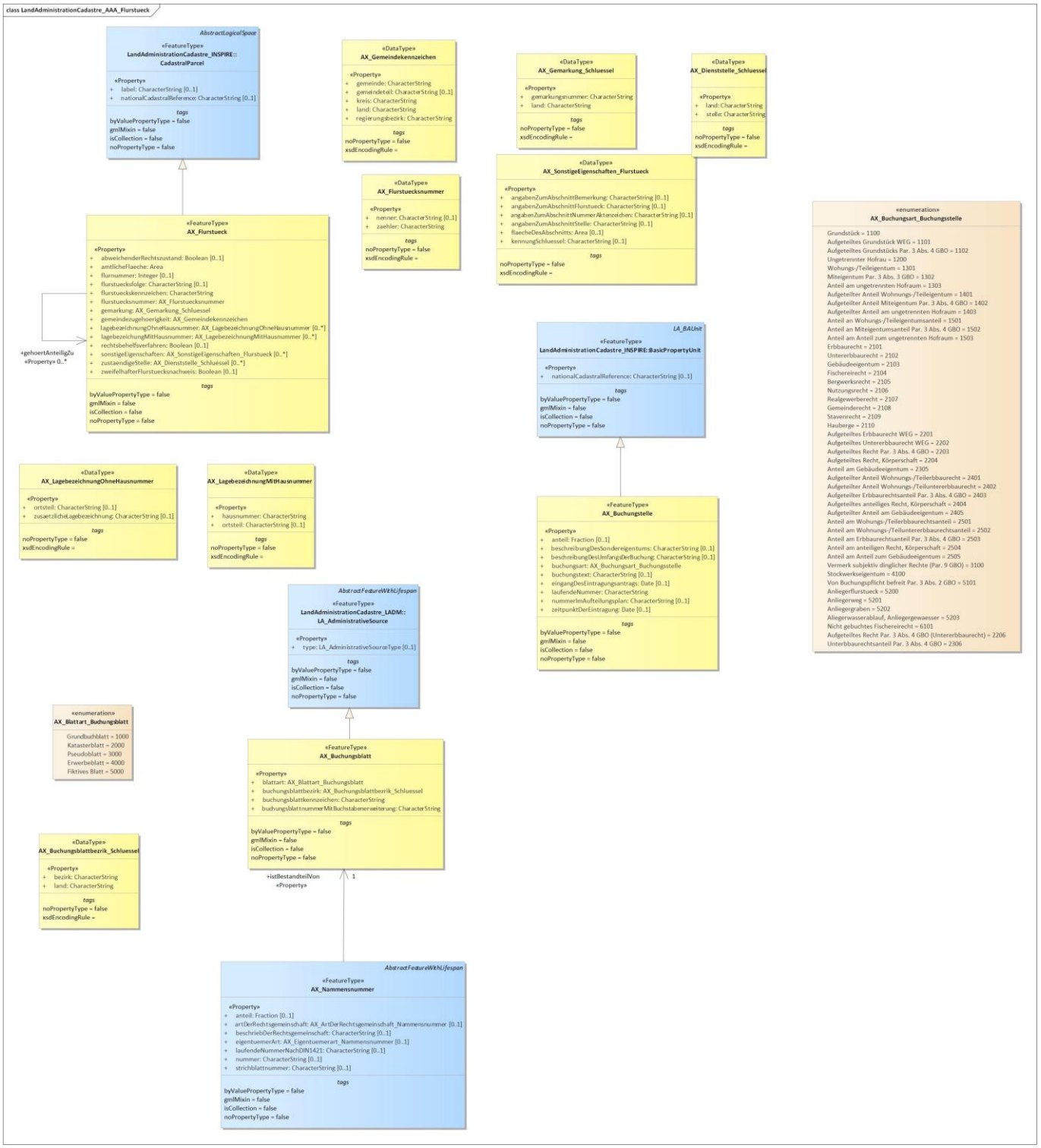


Diagram 28 - LandAdministrationCadastre_AAA_Flurstueck

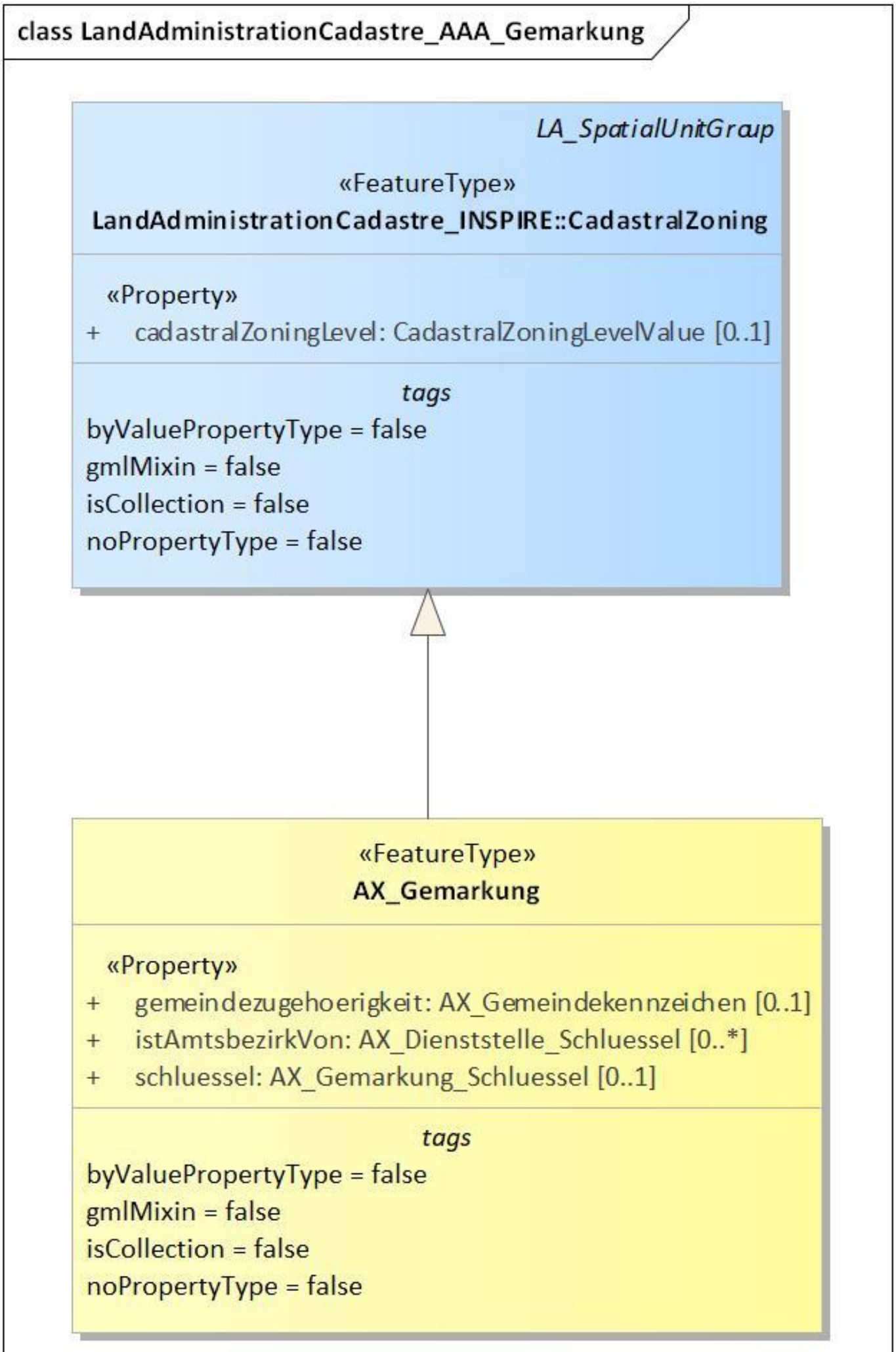
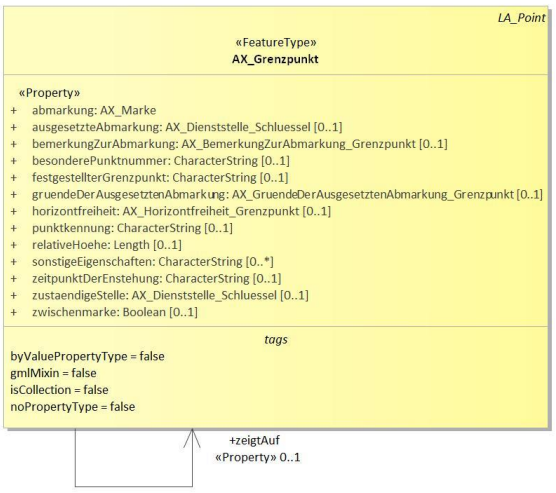

class LandAdministrationCadastre_AAA_Gemarkung
LA_SpatialUnitGroup
«FeatureType»
LandAdministrationCadastre_INSPIRE::CadastralZoning
«Property»
+ cadastralZoningLevel: CadastralZoningLevelValue [0..1]
tags
byValuePropertyType = false
gmlMixin = false
isCollection = false
noPropertyType = false
«FeatureType»
AX_Gemarkung
«Property»
+ gemeindezugehoerigkeit: AX_Gemeindekennzeichen [0..1]
+ istAmtsbezirkVon: AX_Dienststelle_Schluessel [0..*]
+ schluessel: AX_Gemarkung_Schluessel [0..1]
tags
byValuePropertyType = false
gmlMixin = false
isCollection = false
noPropertyType = false

Diagram 29 - LandAdministrationCadastre_AAA_Gemarkung



«enumeration»
AX_GruendeDerAusgesetztenAbmarkung_Grenzpunkt

Grenzpunkt durch bauliche Anlage ausreichend gekennzeichnet = 1000
Grenzpunkt liegt innerhalb einer baulichen Anlage = 2000
Grenzpunkt in oeffentlich-rechtlichem Bodenordungsverfahren = 3000
Grenzpunkt liegt innerhalb eines Baugebietes = 4000
Grenzpunkt liegt innerhalb oder an einem Gewaesser = 5000
Keine Abmarkung aufgrund von anderweitigen Hindernissen = 6000
Abmarkung wuerde unzumutbare Schaeden verursachen = 7000
Langrfistige gemeinschaftliche Nutzung der angrenzenden Flurstuecke = 8000
Angrenzende Flurstuecke dienen dem Gemeingebrauch = 9000
Sonstiges = 9999

«enumeration»
AX_BemerkungZurAbmarkung_Grenzpunkt

Abmarkung unterirdisch gesichert = 1000
Abmarkung exzentrisch gesichert = 2000
Abmarkung uterirdisch und exzentrisch gesichert = 3000
Ohne unterirdische oder exzentrische Sicherung = 4000

«enumeration»
AX_Horizontfreiheit_Grenzpunkt

Uneingeschraenkt = 1000
Eingeschraenkt = 2000
Nicht geeignet = 3000

«enumeration»
AX_Marke

Marke, allgemein = 1000
Stein = 1100
Lochstein = 1111
Vermessungspunktstein = 1112
Unbehauener Feldstein = 1120
Gemeinde- und Waldgrenzstein = 1130
Gemeindegrenzstein = 1131
Waldgrenzstein, Forstgrenzstein = 1132
Kunstoffmarke = 1140
Landesgrenzstein = 1160
Stein mit Besonderheiten in Form oder Material = 1190
Rohr = 1200
Rohr mit Schutzkappe = 1201
Rohr mit Kopf = 1202
Rohr mit Bolzen, oberirdisch = 1203
Eisenrohr = 1210
Eisenrohr (mit Schutzkappe) = 1211
Eisenrohr (ohne Schutzkappe) = 1212
Kunststoffrohr = 1220
Kunststoffrohr (mit Schutzkappe) = 1221
Kunststoffrohr (ohne Schutzkappe) = 1222
Drainrohr = 1230
Rohr mit Schutzkasten = 1240
Zementrohr = 1250
Glasrohr = 1260
Tonrohr = 1290
Bolzen/Nagel = 1300
Bolzen = 1310
Adapterbolzen = 1311
Nagel = 1320
Meisselzeichen (z.B. Kreuz, Kerbe, Anker) = 1400
Bohrloch = 1410
Pfahl = 1500
Sonstige Marke = 1600
Marke in Schutzberhälter = 1610
Flasche = 1620
Platte = 1630
Klinkerplatte = 1631
Granitplatte = 1632
Platte mit Loch = 1635
Hohlziegel = 1640
Klebemarke = 1650
Schlagmarke = 1655
Kanaldeckel (Kreuz des Gütesiegels auf Rand) = 1660
Marke besonderer Ausführung = 1670
Punkt dauerhaft und gut erkennbar festgelegt = 1700
Punkt der baulichen Anlage = 1710
Sockel (roh) = 1711
Sockel (verputzt) = 1712
Mauerecke (roh) = 1713
Mauerecke (verputzt) = 1714
Grenzsaeule = 1720
Pfeiler = 1800
Kegel = 1820
Festlegung 1. Ordnung, Kopf 30x30 cm, Bezugspunkt Platte = 2100
Festlegung 1. Ordnung, Bezugspunkt Kopf 30x30 cm = 2101
Festlegung STN 1. Ordnung, Pfeilerkopf 30x30 cm, Bezugspunkt Platte 60x60 cm, Steinwürfel, Tonkegel = 2102
Festlegung 2. bis 5. Ordnung, Kopf 16x16 oder 12x12 cm, Bezugspuknt Platte 30x30 cm = 2110
Festlegung 2. bis 5. Ordnung, Bezugspunkt Kopf 16x16 oder 12x12 cm, Platte 30x30 cm = 2111
Festlegung 2. bis 4. Ordnung, Kopf 20x20 cm, Bezugspunkt Platte = 2120
Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 20x20 cm = 2121
Festlegung 2. bis 4. Ordnung, Kopf 25x25 cm, Bezugspunkt Platte = 2130
Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 25x25 cm = 2131
Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Platte 30x30-40x40 cm = 2132
Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Bolzen oder Rotgusskappe im Fels = 2133
Festlegung TP-Feld, Pfeilerkopf 25x25 cm, Bezugspunkt Platte 35x35 cm = 2134
Festlegung RBP-Feld, Pfeilerkopf 16x16 cm mit Gravur "TP" und "Dreieck", Bezugspunkt Platte 30x30-35x35 cm = 2135
Plattformbolzen mit Aufschrift TP = 2140
Turmbolzen mit Aufschrift TP = 2150
Leuchtschraube oder -bolzen = 2160
Schraube (vertikal) = 2161
Messingmarke oder Messingbolzen = 2162
Keramikbolzen, oberirdisch = 2163
Bolzen im Fels, unterirdisch = 2164
Rotgusskappe im Fels, unterirdisch = 2165
Messingbolzen (gewölbt), Aufschrift TP und Dreieck = 2166
3D-Messingbolzen (Durchmesser 5 cm) mit Inschrift RFP HESSEN = 2167
Turmbolzen, Festlegungsbolzen oder sonstiger Bolzen, keine weiteren Angaben bekannt oder gespeichert = 2170
Festlegung 2. Ordnung, Kopf 16x16 cm oder 12x12 cm, Bezugspunkt Platte 60x60 cm = 2180
Festlegung 2. Ordnung, Bezugspunkt Kopf 16x16 cm, oder 12x12 cm, Platte 60x60 cm = 2181
Festlegung 2. bis 3. Ordnung, Kopf 16x16 cm oder 12x12 cm, Platte 40x40 cm = 2190
Festlegung 2. bis 3. Ordnung, Bezugspunkt Kopf 16x16 cm, Platte 40x40 cm = 2191
Bergpfeiler ca. 60 cm lang, ohne Platte, Kopf 12x12 cm = 2192
Pfeiler mit Aufschrift AP = 2200
Pfeiler mit Aufschrift AP mit Platte, Bezugspunkt Kopf = 2201
Plattformbolzen mit der Aufschrift AP = 2210
Turmbolzen mit der Aufschrift AP = 2220
Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 1 = 2230
Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 2 = 2240
Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 3 = 2250
Festlegung Sachsen-Anhalt SANREF (unterirdischer Granitpfeiler mit Kopfbolzen) = 2260
3D-Bolzen einzementiert in Kopfflaeche einer Festlegung STN 1. Ordnung = 2261
3D-Bolzen in Kopfflaeche einer Festlegung TP-Feld = 2262
3D-Bolzen in Kopffläche einer AF Sachsen 3. Ordnung = 2263
3D-Bolzen in Kopfflaeche einer AF Sachsen 5. Ordnung = 2264
3D-Bolzen in Granitplatte unter Schutzkasten oder Kunststoff-Schutzrohr = 2265
3D-Bolzen in Kopffläche, einbetonierter Granitpfeiler 40x40x90 cm = 2266
3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 25x25x100 cm = 2267
3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 30x30x100 cm = 2268
3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 50x50x100 cm = 2269
Alte Festlegung der Kgl. Generalkomission und von Kurhessen, Rillenstein = 2300
Alte Festlegung von Nassau, exz. ehemals 2.O. = 2310
Alte Festlegung von Nassau, exz. ehemals 3.O. = 2320
Alte Festlegung von Hessen-Darstadt, 3.O. = 2330
Alte Festlegung von Hessen-Darmstadt, 4.O. = 2340
Alte Festlegung von Hessen-Darmstadt, 3.O. exz. = 2350
Alte Festlegung von Hessen-Darmstadt, 4.O. exz. = 2360
Alte Festlegung von Westfalen, 2. O. exz. = 2370
Alte Festlegung von Württemberg, exz. = 2400
Alte Festlegung von Württemberg = 2410
Alte Festlegung von Mecklenburg, ehemals 1. bis 2. Ordnung = 2420
Alte Festlegung von Mecklenburg 2. bis 3. Ordnung = 2430
Alte Festlegung von Mecklenburg, ehemals 4. Ordnung = 2440
Alte Festlegung Sachsen, 1. und 2. Ordnung (Nagelsche Saeule) = 2450
Alte Festlegung Sachsen, 3. Ordnung, Pfeilerkopf 35x35 cm mit zentrischer Messingmarke = 2460
Alte Festlegung Sachsen, 5. Ordnung, Pfeilerkopf 25x25 cm mit zentrischer Mesingmarke = 2470

Diagram 30 - LandAdministrationCadastre_AAA_Grenzpunkt

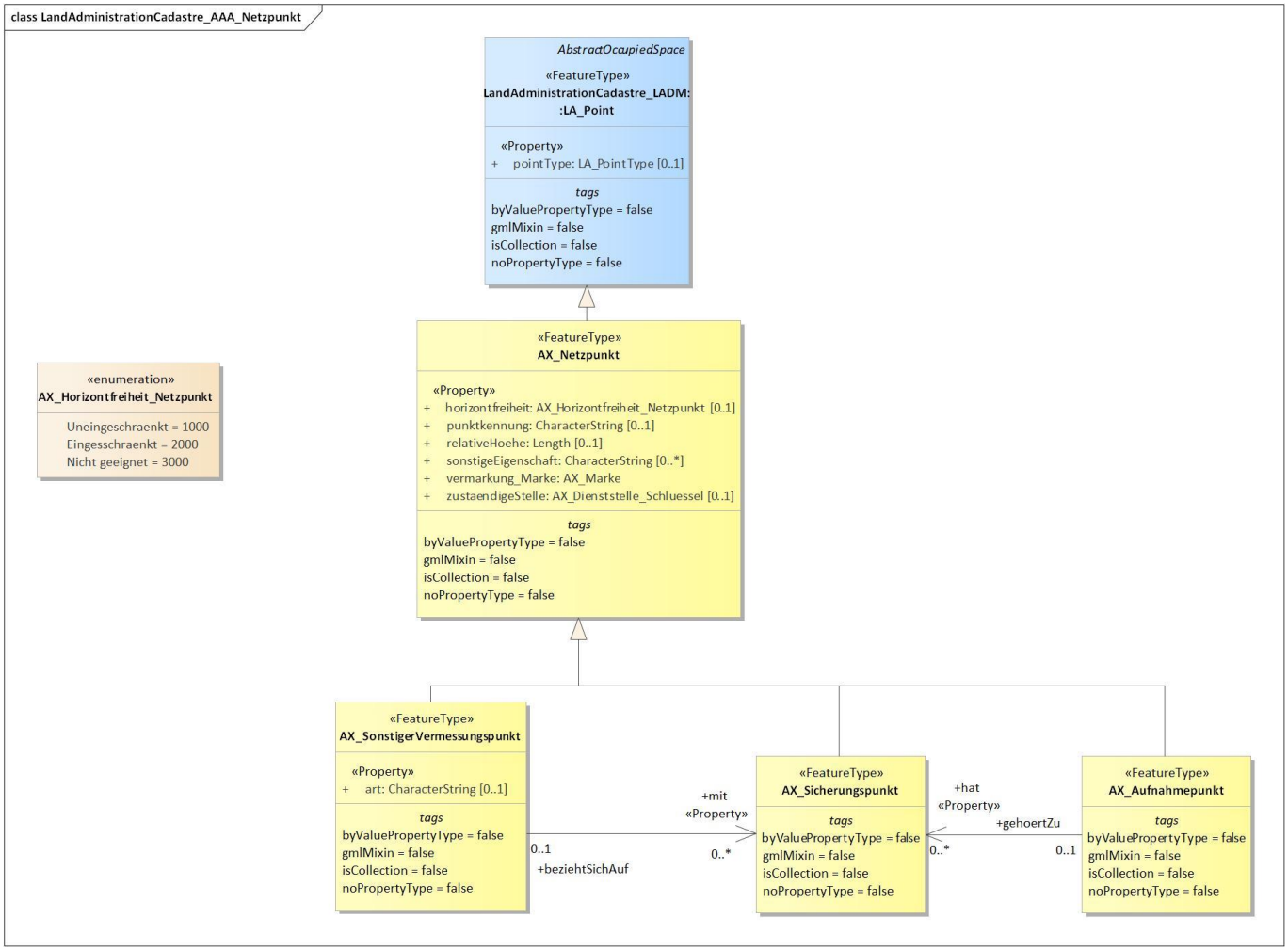


Diagram 31 - LandAdministrationCadastre_AAA_Netzpunkt

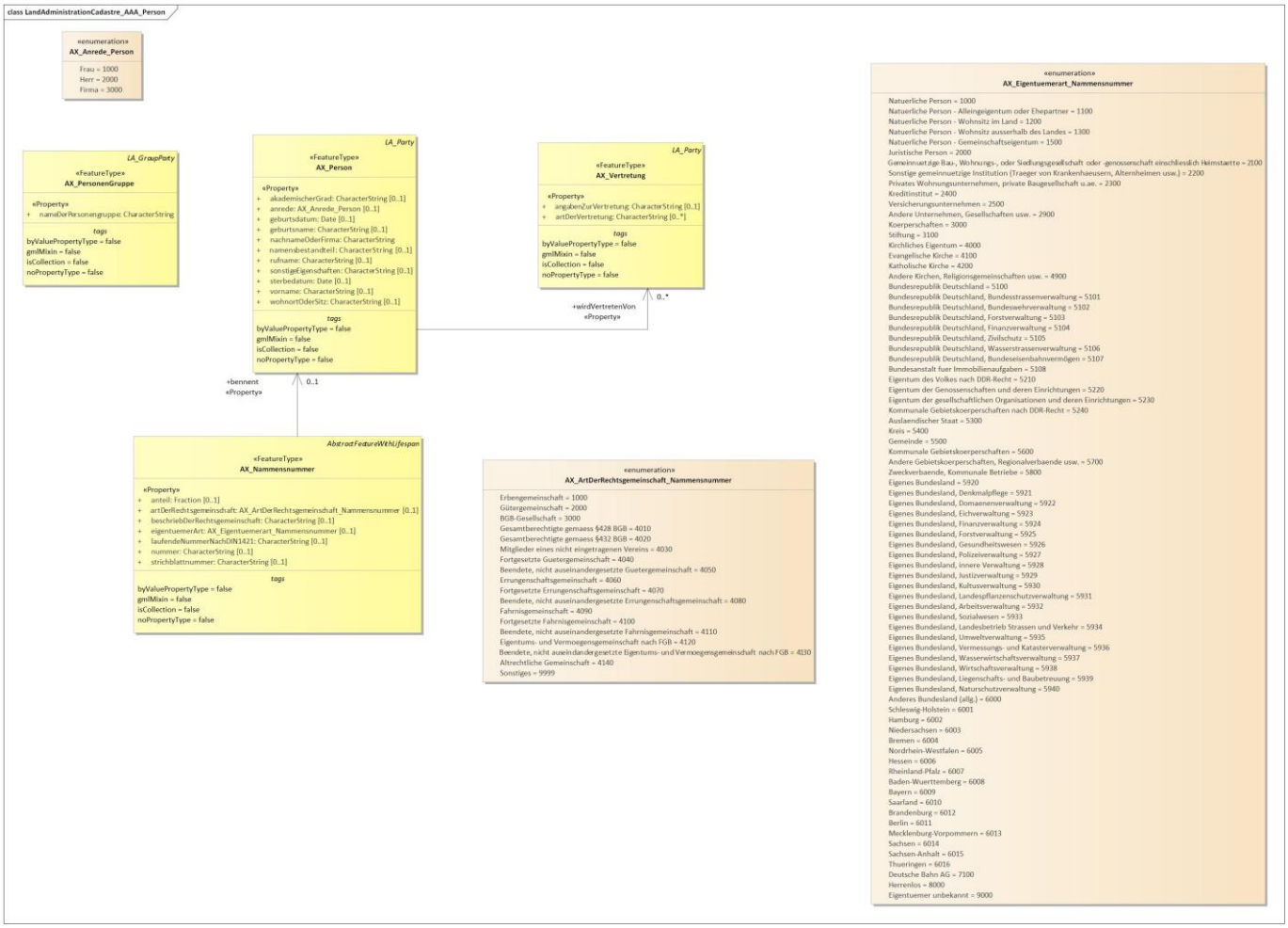


Diagram 32 - LandAdministrationCadastre_AAA_Person

1.9.1 AX_Aufnahmepunkt

AX_Aufnahmepunkt	
Definition:	From the AAA 7.1 specification: 'Aufnahmepunkt' ist ein Punkt des Lagefestpunktfeldes - Aufnahmepunktfeld und dient der örtlichen Aufnahme von Objektpunkten. Das Aufnahmepunktfeld ist eine Verdichtungsstufe des Lagefestpunktfeldes - Trigonometrisches Festpunktfeld (Grundlagenvermessung).
Subtype of:	AX_Netzpunkt
Type:	Feature type
Association role	
Name:	hat
Voidable:	false
Multiplicity:	0..*
Value type:	AX_Sicherungspunkt (feature type)

1.9.2 AX_BesondereFlurstuecksgrenze

AX_BesondereFlurstuecksgrenze

Definition: From the AAA 7.1 specification:

'Besondere Flurstücksgrenze' ist ein Teil der Grenzlinie eines Flurstücks, der von genau zwei benachbarten Grenzpunkten begrenzt wird und für den besondere Informationen vorliegen.

Subtype of: CadastralBoundary

Type: Feature type

Attribute:

Name: artDerFurstuecksgrenze

Definition: From the AAA 7.1 specification:

'Art der Flurstücksgrenze ' ist die Benennung der besonderen Information zur Flurstücksgrenze.

Es sind jeweils alle Funktionen, die eine Flurstücksgrenze in sich vereinigt, auch explizit zu führen.

Voidable: false

Multiplicity: 1..*

Value type: AX_ArtDerFlurstuecksgrenze_BesondereFlurstuecksgrenze (enumeration)

Values

1000	**Strittige Grenze**
2001	**Nicht festgestellte Grenze**
2002	**Verwaltungsstreitverfahren (Grenze)**
2003	**Mittellinie in Gewaessern**
2004	**Flurstuecksgrenze nicht feststellbar**
2010	**Topographische Grenzeinrichtung**
2100	**Topographische Gewaesserbegrenzung**
2500	**Grenze der Region**
3000	**Grenze der Flur**
7003	**Grenze der Gemarkung**
7101	**Grenze der Bunderepublik Deutschland**
7102	**Grenze des Bundeslandes**

	7103	**Grenze des Regierungsbezirks**
	7104	**Grenze des Landkreises**
	7106	**Grenze der Gemeinde**
	7107	**Grenze des Gemeindeteils**
	7108	**Grenze der Verwaltungsgemeinschaft**

1.9.3 AX_Bodenschaetzung

AX_Bodenschaetzung	
Definition:	From the AAA 7.1 specification: Bodenschätzung ist die kleinste Einheit einer bodengeschätzten Fläche nach dem BodSchätzG, für die eine Ertragsfähigkeit im Liegenschaftskataster nachzuweisen ist (Bodenschätzungsfläche). Die Attributarten 'Nutzungsart' und 'Bodenart' sind objektbildend. Wird eine Bodenschätzungsfläche durch eine Fläche, die nicht Bodenschätzungsfläche ist durchschnitten (z.B. Straße, Weg, Gewässer), kann die Modellierung auf der Grundlage von zwei oder mehr getrennt liegenden Flächen erfolgen.
Subtype of:	DerivedSoilProfile
Type:	Feature type
Attribute:	
Name:	ackerzahlOderGruenlandzahl
Definition:	From the AAA 7.1 specification: 'Ackerzahl oder Grünlandzahl' ist die 'Bodenzahl oder Grünlandgrundzahl' einschließlich Ab- und Zurechnungen nach dem Bodenschätzungsgesetz.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	bodenart
Definition:	From the AAA 7.1 specification: 'Bodenart' ist die nach den Verwaltungsanweisungen zum Bodenschätzungsgesetz festgelegte Bezeichnung der Bodenart.
Voidable:	false

Multiplicity: 1

Value type: AX_Bodenart_Bodenschaetzung (enumeration)

Values

1100	**Sand (S)**
1200	**Anlehmiger Sand (Sl)**
2100	**Lehmiger Sand (lS)**
2200	**Stark lehmiger Sand (SL)**
3100	**Lehm (L)**
3200	**Sandiger Lehm (sL)**
4100	**Ton (T)**
4200	**Schwerer Lehm (LT)**
5000	**Moor (Mo)**
6110	**Sand mit Moor (SMo)**
6120	**Lehmiger Sand mit Moor (lSMo)**
6130	**Lehm mit Moor (LMo)**
6140	**Ton mit Moor (TMo)**
6210	**Moor mit Sand (MoS)**
6220	**Moor mit lehmigem Sand (MolS)**
6230	**Moor mit Lehm (MoL)**
6240	**Moor mit Ton (MoT)**
7100	**Sand auf stark lehmigem Sand (S/SL)**
7110	**Sand auf sandigem Lehm (S/sL)**
7120	**Sand auf Lehm (S/L)**
7130	**Sand auf schwerem Lehm (S/LT)**
7140	**Sand auf Ton (S/T)**
7200	**Anlehmiger Sand auf sandigem Lehm (Sl/sL)**
7210	**Anlehmiger Sand auf Lehm (Sl/L)**
7220	**Anlehmiger Sand auf schwerem Lehm (Sl/LT)**
7230	**Anlehmiger Sand auf Ton (Sl/T)**
7300	**Lehmiger Sand auf Lehm (lS/L)**
7310	**Lehmiger Sand auf schwerem Lehm (lS/LT)**

	7320	**Lehmiger Sand auf Sand (lS/S)**
	7330	**Lehmiger Sand auf Ton (lS/T)**
	7400	**Stark lehmiger Sand auf Ton (SL/T)**
	7410	**Stark lehmiger Sand auf schwerem Lehm (SL/LT)**
	7420	**Stark lehmiger Sand auf Sand (SL/S)**
	7500	**Ton auf sandigem Lehm (T/sL)**
	7510	**Ton auf stark lehmigem Sand (T/SL)**
	7520	**Ton auf lehmigem Sand (T/lS)**
	7530	**Ton auf anlehmigem Sand (T/Sl)**
	7540	**Ton auf Sand (T/S)**
	7600	**Schwerer Lehm auf stark lehmigem Sand (LT/SL)**
	7610	**Schwerer Lehm auf lehmigem Sand (LT/lS)**
	7620	**Schwerer Lehm auf anlehmigem Sand (LT/Sl)**
	7630	**Schwerer Lehm auf Sand (LT/S)**
	7700	**Lehm auf lehmigem Sand (L/lS)**
	7710	**Lehm auf anlehmigem Sand (L/Sl)**
	7720	**Lehm auf Sand (L/S)**
	7800	**Sandiger Lehm auf Sand (sl/S)**
	7810	**Sandiger Lehm auf anlehmigem Sand (sL/Sl)**
	7820	**Sandiger Lehm auf Ton (sL/T)**
	8110	**Sand auf Moor (S/Mo)**
	8120	**Lehmiger Sand auf Moor (lS/Mo)**
	8130	**Lehm auf Moor (L/Mo)**
	8140	**Ton auf Moor (T/Mo)**
	8210	**Moor auf Sand (Mo/S)**
	8220	**Moor auf lehmigem Sand (Mo/lS)**
	8230	**Moor auf Lehm (Mo/L)**
	8240	**Moor auf Ton (Mo/T)**

9120	**Lehm und Moor, Bodenwechsel (L+Mo)**
9130	**Lehmiger Sand, steinig (lSg)**
9140	**Lehm, steinig (Lg)**
9150	**Lehmiger Sand uns Steine/Blöcke (lS+St)**
9160	**Lehm und Steine/Blöcke (L+St)**
9170	**Steine/Blöcke und lehmiger Sand (St+lS)**
9180	**Steine/Blöcke und Lehm (St+L)**
9190	**Lehmiger Sand und Felsen (lS+Fe)**
9200	**Lehm und Felsen (L+Fe)**
9210	**Felsen und lehmiger Sand (Fe+lS)**
9220	**Felsen und Lehm (Fe+L)**
9310	**Sand auf lehmigem Sand (S/lS)**
9320	**Anlehmiger Sand auf Mergel (Sl/Me)**
9350	**Lehmiger Sand auf Mergel (lS/Me)**
9360	**Lehmiger Sand auf sandigem Lehm (lS/sL)**
9370	**Lehmiger Sand mit Mergel (lSMe)**
9380	**Lehmiger Sand mit Moor auf Mergel (lSMo/Me)**
9390	**Anlehmiger Sand mit Moor (SlMo)**
9410	**Lehm auf Mergel (L/Me)**
9420	**Lehm mit Moor auf Mergel (LMo/Me)**
9430	**Schwerer Lehm auf Moor (LT/Mo)**
9440	**Ton auf Mergel (T/Me)**
9450	**Moor auf Mergel (Mo/Me)**
9460	**Moor mit Lehm auf Mergel (MoL/Me)**
9470	**Moor mit Mergel (MoMe)**
9480	**LößDiluvium (LöD)**
9490	**AlluviumDiluvium (AlD)**

Attribute:

Name: bodenstufe

Definition: From the AAA 7.1 specification:

'Bodenstufe' ist die nach dem Grünlandschätzungsrahmen festgelegte Bezeichnung der Bodenstufe.

Voidable: false

Multiplicity: 0..1

Value type: AX_Bodenstufe (enumeration)

Values

2100	**Bodenstufe (I)**
2200	**Bodenstufe (II)**
2300	**Bodenstufe (III)**
2400	**Bodenstufe Misch- und Schichtböden sowie künstlich veränderte Böden (-)**
3100	**Bodenstufe (II+III)**
3200	**Bodenstufe ("(III)")**
3300	**Bodenstufe (IV)**

Attribute:

Name: bodenzahlOderGruenlandgrundzahl

Definition: From the AAA 7.1 specification:

'Bodenzahl oder Grünlandgrundzahl' ist die Wertzahl nach dem Acker- oder Grünlandschätzungsrahmen.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: entstehungsart

Definition: From the AAA 7.1 specification:

'Entstehungsart' ist die nach dem Ackerschätzungsrahmen festgelegte Bezeichnung der Entstehungsart.

Voidable: false

Multiplicity: 0..*

Value type: AX_Entstehungsart (enumeration)

Values

1000	**Diluvium (D)**
1100	**Diluvium, Alluvium (DAl)**
1110	**Diluvium, Alluvium, grob, steinig (DAlg)**

1200	**Diluvium, Löß (DLö)**
1300	**Diluvium, Verwitterung (DV)**
1310	**Diluvium, Verwitterung, grob, steinig (DVg)**
1400	**Diluvium, grob, steinig (Dg)**
1410	**Diluvium, grob, steinig, Alluvium (DgAl)**
1420	**Diluvium, grob, steinig, Löß (DgLö)**
1430	**Diluvium, grob, steinig, Verwitterung (DgV)**
2000	**Löß (Lö)**
2100	**Löß, Dilivium (LöD)**
2110	**Löß, Dilivium, grob, steinig (LöDg)**
2120	**Löß, Dilivium, Verwitterung (LöDV)**
2200	**Löß, Alluvium (LöAl)**
2210	**Löß, Alluvium, grob, steinig (LöVg)**
2300	**Löß, Verwitterung (LöV)**
2310	**Löß, Verwitterung, grob, steinig (LöVg)**
2400	**Löß über Verwitterung, gesteinig (LöVg)**
3000	**Alluvium (Al)**
3100	**Alluvium, Diluvium (AlD)**
3200	**Alluvium, Löß (AlLö)**
3300	**Alluvium, Verwitterung (AlV)**
3310	**Alluvium, Verwitterung, grob, steinig (AlVg)**
3400	**Alluvium, grob, steinig (Alg)**
3410	**Alluvium, grob, steinig, Dilivium (AlgD)**
3430	**Alluvium, grob, steinig, Verwitterung (AlgV)**
3420	**Alluvium, grob, steinig, Löß (AlgLö)**
3500	**Alluvium, Marsch (AlMa)**
3610	**Alluvium, Moor (AlMo)**

3620	**Moor, Alluvium (MoAl)**
3700	**Mergel (Me)**
4000	**Verwitterung (V)**
4100	**Verwitterung, Diluvium (VD)**
4110	**Verwitterung, Diluvium, grob, steinig (VDg)**
4200	**Verwitterung, Alluvium (VAl)**
4210	**Verwitterung, Alluvium, grob, steinig (VAlg)**
4300	**Verwitterung, Löß (VLö)**
4400	**Verwitterung, grob, steinig (Vg)**
4410	**Verwitterung, grob, steinig, Diluvium (VgD)**
4420	**Verwitterung, grob, steinig, Löß (VgLö)**
4430	**Verwitterung, grob, steinig, Alluvium (VgAl)**
5000	**Entstehungsart nicht erkennbar (-)**

Attribute:

Name: jahreszahl

Definition: From the AAA 7.1 specification:

'Jahreszahl' ist das Jahr, in dem eine Neukultur oder Tiefkultur erstmals in Kultur genommen worden ist.

Voidable: false

Multiplicity: 0..1

Value type: Integer

Attribute:

Name: klimastufe

Definition: From the AAA 7.1 specification:

'Klimastufe' ist die nach dem Grünlandschätzungsrahmen festgelegte Bezeichnung der Klimastufe.

Voidable: false

Multiplicity: 0..1

Value type: AX_Klimastufe (enumeration)

Values

6000	**Klimastufe nicht erkennbar (-)**

6100	**Klimastufe 8° C und darüber (a)**
6200	**Klimastufe 7,9° -7,0° C (b)**
6300	**Klimastufe 6,9° - 5,7° C (c)**
6400	**Klimastufe 5,6° C und darunter (d)**

Attribute:

Name: nuztungsart

Definition: From the AAA 7.1 specification:

'Nutzungsart' ist die bestandskräftig festgesetzte landwirtschaftliche Nutzungsart entsprechend dem Acker- oder Grünlandschätzungsrahmen.

Voidable: false

Multiplicity: 1

Value type: AX_Nutzungsart_Bodenschaetzung (enumeration)

Values

1000	**Ackerland (A)**
2000	**Acker-Grünland (AGr)**
3000	**Grünland (Gr)**
4000	**Grünland-Acker**

Attribute:

Name: sonstigeAngaben

Definition: From the AAA 7.1 specification:

'Sonstige Angaben' ist der Nachweis von Besonderheiten einer bodengeschätzten Fläche.

Voidable: false

Multiplicity: 0..*

Value type: AX_SonstigeAngaben_Bodenschaetzung (enumeration)

Values

1100	**Nass, zu viel Wasser (Wa+)**
1200	**Trocken, zu wenig Wasser (Wa-)**
1300	**Besonders günsitge Wasserverhältnisse (Wa gt)**
1400	**Rieselwasser, künstliche Bewässerung (RiWa)**
2100	**Unbedingtes Wiesenland (W)**
2200	**Streuwiese (Str)**
2300	**Hutung (Hu)**

2400	**Acker-Hackrain (Hack)**
2500	**Grünland-Hackrain (Hack)**
2600	**Garten (G)**
3000	**Neukultur (N)**
4000	**Tiefkultur**
5000	**Geringsland (Ger)**
9000	**Nachschätzung erforderlich**

Attribute:

Name: wasserverhaeltnisse

Definition: From the AAA 7.1 specification:

'Wasserverhältnisse' ist die nach dem Grünlandschätzungsrahmen festgelegte Bezeichnung der Wasserverhältnisse.

Voidable: false

Multiplicity: 0..1

Value type: AX_Wasserverhaeltnisse (enumeration)

Values

7000	**Wasserstufe nicht erkennbar (-)**
7100	**Wasserstufe (1)**
7200	**Wasserstufe (2)**
7300	**Wasserstufe (3)**
7400	**Wasserstufe (4)**
7310	**Wasserstufe (3-)**
7410	**Wasserstufe (4-)**
7500	**Wasserstufe (5)**
7510	**Wasserstufe (5-)**
7600	**Wasserstufe (3+4)**

Attribute:

Name: zustandsstufe

Definition: From the AAA 7.1 specification:

'Zustandsstufe' ist die nach dem Ackerschätzungsrahmen festgelegte Bezeichnung der Zustandsstufe.

Voidable: false

Multiplicity: 0..1

Value type:	AX_Zustandsstufe (enumeration)
Values	

1100	**Zustandsstufe (1)**
1200	**Zustandsstufe (2)**
1300	**Zustandsstufe (3)**
1400	**Zustandsstufe (4)**
1500	**Zustandsstufe (5)**
1600	**Zustandsstufe (6)**
1700	**Zustandsstufe (7)**
1800	**Zustandsstufe Misch- und Schichtböden sowie künstlich veränderte Böden (-)**

1.9.4 AX_Buchungsblatt

AX_Buchungsblatt

Definition:	From the AAA 7.1 specification: 'Buchungsblatt' enthält die Buchungen (Buchungsstellen und Namensnummern) des Grundbuchs und des Liegenschaftskatasters (bei buchungsfreien Grundstücken). Das Buchungsblatt für Buchungen im Liegenschaftskataster kann entweder ein Kataster-, Erwerber-, Pseudo- oder ein Fiktives Blatt sein.
Subtype of:	LA_AdministrativeSource
Type:	Feature type

Attribute:

Name:	blattart
Definition:	From the AAA 7.1 specification: 'Blattart' ist die Art des Buchungsblattes.
Voidable:	false
Multiplicity:	1
Value type:	AX_Blattart_Buchungsblatt (enumeration)
Values	

1000	**Grundbuchblatt**
2000	**Katasterblatt**
3000	**Pseudoblatt**
4000	**Erwerbeblatt**
5000	**Fiktives Blatt**

Attribute:	
Name:	buchungsblattbezirk
Definition:	From the AAA 7.1 specification: Buchungsblattbezirk des Buchungsblatts.
Voidable:	false
Multiplicity:	1
Value type:	AX_Buchungsblattbezrik_Schluessel (data type)
Attribute:	
Name:	buchungsblattkennzeichen
Definition:	From the AAA 7.1 specification:'Buchungsblattkennzeichen' ist ein eindeutiges Fachkennzeichen für ein Buchungsblatt. Aufbau Buchungsblattkennzeichen: 1.) Land (Verschlüsselung zweistellig), 2 Ziffern 2.) Buchungsblattbezirk (Verschlüsselung vierstellig), 4 Ziffern 3.) Buchungsblattnummer mit Buchstabenerweiterung (7 Stellen) Die Elemente sind rechtsbündig zu belegen, fehlende Stellen sind mit führenden Nullen zu belegen. Die Gesamtlänge des Buchungsblattkennzeichens beträgt immer 13 Zeichen
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	buchungsblattnummerMitBuchstabenerweiterung
Definition:	From the AAA 7.1 specification: Buchungsblattnummer mit Buchstabenerweiterung.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.5 AX_Buchungsblattbezrik_Schluessel

AX_Buchungsblattbezrik_Schluessel	
Definition:	From the AAA 7.1 specification: Amtliche Verschlüsselung des Buchungsblattbezirks.

Type:	Data type
Attribute:	
Name:	bezirk
Definition:	From the AAA 7.1 specification: Kennung des Bezirks.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: Kennung des Bezirks.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.6 AX_Buchungstelle

AX_Buchungstelle	
Definition:	From the AAA 7.1 specification: 'Buchungsstelle' ist die unter einer laufenden Nummer im Verzeichnis des Buchungsblattes eingetragene Buchung. Die Buchungsarten mit Wertearten 1101, 1102, 1401 bis 1403, 2201 bis 2205 und 2401 bis 2405 können nur auf einem Fiktiven Blatt vorkommen.Die Attributart 'Anteil' ist optional zu belegen, sofern konkrete und in sich schlüssige Angaben hierzu vorliegen.
Subtype of:	BasicPropertyUnit
Type:	Feature type
Attribute:	
Name:	anteil
Definition:	From the AAA 7.1 specification: 'Anteil' ist die Angabe des Miteigentumsanteils am Grundstück oder des Anteils am Recht. Das Attribut setzt sich zusammen aus: 1. Spalte: Zähler

	2. Spalte: Nenner
Voidable:	false
Multiplicity:	0..1
Value type:	Fraction (data type)
Attribute:	
Name:	beschreibungDesSondereigentums
Definition:	From the AAA 7.1 specification: 'Beschreibung des Sondereigentums' ist die Beschreibung von Wohnungseigentum an Wohnungen und von Teileigentum an nicht zu Wohnzwecken dienenden Räumen. Die Information wird nach Einführung des Datenbankgrundbuches (DaBaG) von der Grundbuchverwaltung nicht mehr übermittelt.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	beschreibungDesUmfangsDerBuchung
Definition:	From the AAA 7.1 specification: 'Beschreibung des Umfangs der Buchung' ist eine nähere Beschreibung der Buchungsart (z.B. 'von der Quelle bis zur Brücke').
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	buchungsart
Definition:	From the AAA 7.1 specification: 'Buchungsart' bezeichnet die Art der Buchung. Die Werte 1200, 1303, 1401, 1402, 1403, 1501, 1502, 1503, 2105, 2107, 2108, 2110, 2204, 2304, 2401, 2402, 2403, 2404, 2405, 2501, 2502, 2503, 2504, 2505, 3100 und 6101 werden nach Einführung des Datenbankgrundbuches (DaBaG) von der Grundbuchverwaltung nicht mehr übermittelt.
Voidable:	false

Multiplicity: 1

Value type: AX_Buchungsart_Buchungsstelle (enumeration)

Values

1100	**Grundstück**
1101	**Aufgeteiltes Grundstück WEG**
1102	**Aufgeteiltes Grundstücks Par. 3 Abs. 4 GBO**
1200	**Ungetrennter Hofrau**
1301	**Wohungs-/Teileigentum**
1302	**Miteigentum Par. 3 Abs. 3 GBO**
1303	**Anteil am ungetrennten Hofraum**
1401	**Aufgeteilter Anteil Wohnungs-/Teileigentum**
1402	**Aufgeteilter Anteil Miteigentum Par. 3 Abs. 4 GBO**
1403	**Aufgeteilter Anteil am ungetrennten Hofraum**
1501	**Anteil an Wohungs-/Teileigentumsanteil**
1502	**Anteil an Miteigentumsanteil Par. 3 Abs. 4 GBO**
1503	**Anteil am Anteil zum ungetrennten Hofraum**
2101	**Erbbaurecht**
2102	**Untererbbaurecht**
2103	**Gebäudeeigentum**
2104	**Fischereirecht**
2105	**Bergwerksrecht**
2106	**Nutzungsrecht**
2107	**Realgewerberecht**
2108	**Gemeinderecht**
2109	**Stavenrecht**
2110	**Hauberge**
2201	**Aufgeteiltes Erbbaurecht WEG**
2202	**Aufgeteiltes Untererbbaurecht WEG**

2203	**Aufgeteiltes Recht Par. 3 Abs. 4 GBO**
2204	**Aufgeteiltes Recht, Körperschaft**
2305	**Anteil am Gebäudeeigentum**
2401	**Aufgeteilter Anteil Wohnungs-/Teilerbbaurecht**
2402	**Aufgeteilter Anteil Wohnungs-/Teiluntererbbaurecht**
2403	**Aufgeteilter Erbbaurechtsanteil Par. 3 Abs. 4 GBO**
2404	**Aufgeteiltes anteiliges Recht, Körperschaft**
2405	**Aufgeteilter Anteil am Gebäudeeigentum**
2501	**Anteil am Wohungs-/Teilerbbaurechtsanteil**
2502	**Anteil am Wohnungs-/Teiluntererbbaurechtsanteil**
2503	**Anteil am Erbbaurechtsanteil Par. 3 Abs. 4 GBO**
2504	**Anteil am anteiligen Recht, Körperschaft**
2505	**Anteil am Anteil zum Gebäudeeigentum**
3100	**Vermerk subjektiv dinglicher Rechte (Par. 9 GBO)**
4100	**Stockwerkseigentum**
5101	**Von Buchungspflicht befreit Par. 3 Abs. 2 GBO**
5200	**Anliegerflurstueck**
5201	**Anliegerweg**
5202	**Anliegergraben**
5203	**Aliegerwasserablauf, Anliegergewaesser**
6101	**Nicht gebuchtes Fischereirecht**
2206	**Aufgeteiltes Recht Par. 3 Abs. 4 GBO (Untererbbaurecht)**

	2306	**Unterbbaurechtsanteil Par. 3 Abs. 4 GBO**

Attribute:

Name:	buchungstext
Definition:	From the AAA 7.1 specification: 'Buchungstext' enthält zusätzliche Angaben zur Buchungsart (z.B. die genaue Bezeichnung von Nutzungsrechten). Die Information wird nach Einführung des Datenbankgrundbuches (DaBaG) von der Grundbuchverwaltung nicht mehr übermittelt.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	eingangDesEintragungsantrags
Definition:	From the AAA 7.1 specification: 'Eingang des Eintragungsantrags' gibt das Eingangsdatum für den Antrag auf Veränderung eines Wohnungseigentums, Teileigentums, Erbbaurechts, Wohnungserbbaurechts oder Teilerbbaurechts in der Grundbuchverwaltung an.
Voidable:	false
Multiplicity:	0..1
Value type:	Date

Attribute:

Name:	laufendeNummer
Definition:	From the AAA 7.1 specification: 'Laufende Nummer' ist die eindeutige Nummer der Buchungsstelle auf dem Buchungsblatt.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

Attribute:

Name:	nummerImAufteilungsplan
Definition:	From the AAA 7.1 specification: 'Nummer im Aufteilungsplan' ist die Nummer entsprechend der Teilungserklärung über die Aufteilung des Gebäudes in

	Lage und Größe der im Sondereigentum und der im gemeinschaftlichen Eigentum stehenden Gebäudeteile.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zeitpunktDerEintragung
Definition:	From the AAA 7.1 specification: 'Zeitpunkt der Eintragung' beinhaltet das Datum, an dem die Rechtsänderung stattgefunden hat (z.B. Eintragung im Grundbuch).
Voidable:	false
Multiplicity:	0..1
Value type:	Date

1.9.7 AX_Bundesland_Schluessel

AX_Bundesland_Schluessel	
Definition:	From the AAA 7.1 specification: Amtliche Verschlüsselung des Bundeslands.
Type:	Data type
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: Bundesland.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.8 AX_DQFestpunkt

AX_DQFestpunkt	
Definition:	From the AAA 7.1 specification: Qualitätsangaben zu den Daten des Festpunkts.
Type:	Data type
Attribute:	

Name: befund

Definition: From the AAA 7.1 specification:

In 'Befund' wird der Zustand des Punktes bei der letzten örtlichen Kontrolle sowie die evtl. getroffene Maßnahme angegeben.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: gnssTauglichkeit

Definition: From the AAA 7.1 specification:

'GNSS-Tauglichkeit' (GNSS = Global Navigation Satellite System) beschreibt die zu erwartenden bzw. nachgewiesenen Empfangsmöglichkeiten bei Satellitenmessverfahren.

Voidable: false

Multiplicity: 0..1

Value type: AX_GNSSTauglichkeit (enumeration)

Values

1000	**Vermutlich sehr gut**
1001	**Sehr gut**
3000	**Vermutlich gut**
3001	**Gut**
3100	**Vermutlich befriedigend**
31001	**Befriedigend**
5000	**Ungenügend**
9998	**Nicht untersucht**

Attribute:

Name: punktstabilitaet

Definition: From the AAA 7.1 specification:

'Punktstabilität' gibt die vermutete bzw. nachgewiesene Lage- und ggf. Höhenstabilität der Vermarkung an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet (enumeration)

Values	1000	**Sehr gut**
	2000	**Gut**
	3000	**Befriedigend**
	4000	**Ausreichend**
	5000	**Mangelhaft**
	5100	**Mangelhaft (Bergesenkungsgebiet)**
	5200	**Mangelhaft (in rutschgefaehrdeter Hanglage)**
	5300	**Mangelhaft (sehr nahe an Gewaesser)**
	5400	**Mangelhaft (instabiler Untergrund)**
	6000	**Aus Wiederholungsmessungen nachgewiesen**
	9998	**Nicht untersucht**

Attribute:

Name: ueberwachungsdatum

Definition: From the AAA 7.1 specification:

In 'Überwachungsdatum' wird das Datum der letzten Kontrolle des örtlichen Zustandes des Punktes angegeben.

Voidable: false

Multiplicity: 0..1

Value type: Date

1.9.9 AX_DQHoehenfestpunkt

AX_DQHoehenfestpunkt

Definition: From the AAA 7.1 specification:

Qualitätsangaben zu den Daten des Höhenfestpunkts.

Type: Data type

Attribute:

Name: befund

Definition: From the AAA 7.1 specification:

In 'Befund' wird der Zustand des HFP bei der letzten örtlichen Kontrolle sowie die evtl. getroffene Maßnahme angegeben.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: gnssTauglichkeit

Definition: From the AAA 7.1 specification:

'GNSS-Tauglichkeit' (GNSS = Global Navigation Satellite System) beschreibt die vermuteten bzw. nachgewiesenen Empfangsmöglichkeiten bei Satellitenmessverfahren.

Voidable: false

Multiplicity: 0..1

Value type: AX_GNSSTauglichkeit (enumeration)

Values

1000	**Vermutlich sehr gut**
1001	**Sehr gut**
3000	**Vermutlich gut**
3001	**Gut**
3100	**Vermutlich befriedigend**
31001	**Befriedigend**
5000	**Ungenügend**
9998	**Nicht untersucht**

Attribute:

Name: punktstabilitaet

Definition: From the AAA 7.1 specification:

'Punktstabilität' gibt die vermutete bzw. nachgewiesene Höhenstabilität der Punktvermarkung an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt (data type)

Attribute:

Name: ueberwachungsdatum

Definition: From the AAA 7.1 specification:

In 'Überwachungsdatum' wird das Datum der letzten Kontrolle des örtlichen Zustandes des HFP angegeben.

Voidable: false

Multiplicity: 0..1

Value type:	Date

1.9.10 AX_Dienststelle_Schluessel

AX_Dienststelle_Schluessel	
Definition:	From the AAA 7.1 specification: Amtliche Verschlüsselung der Dienststelle.
Type:	Data type
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: 'Dienststelle' liegt innerhalb eines 'Bundeslandes'.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	stelle
Definition:	From the AAA 7.1 specification: Schlüssel der Dienststelle im Bundesland.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.11 AX_Festpunkt

AX_Festpunkt	
Definition:	From the AAA 7.1 specification: Oberklasse aller Festpunkte.
Subtype of:	LA_Point
Supertype of:	AX_Hoehenfestpunkt AX_Lagefestpunkt
Type:	Feature type
Attribute:	
Name:	erstvermarkung
Definition:	From the AAA 7.1 specification:

	'Erstvermarkung' gibt das Datum der erstmaligen Vermarkung eines Punktes an.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	frueherePunktNummer
Definition:	From the AAA 7.1 specification: Unter 'Frühere Punktnummer' können weitere Punktnummern angegeben werden, die der Festpunkt früher einmal hatte.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString
Attribute:	
Name:	gemarkung
Definition:	From the AAA 7.1 specification: 'Gemarkung' ist die Gemarkung, in welcher der Festpunkt liegt.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Gemarkung_Schluessel (data type)
Attribute:	
Name:	gemeinde
Definition:	From the AAA 7.1 specification: 'Gemeinde' ist die politische Gemeinde, in welcher der Festpunkt liegt.
Voidable:	false
Multiplicity:	1
Value type:	AX_Gemeindekennzeichen (data type)
Attribute:	
Name:	interneBemerkungen
Definition:	From the AAA 7.1 specification: 'Interne Bemerkungen' enthält Bemerkungen zu dem Festpunkt für den internen Dienstbetrieb.

Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString
Attribute:	
Name:	katasteramt
Definition:	From the AAA 7.1 specification: 'Katasteramt' verweist auf die katasterführende Stelle, in deren Amtsbezirk der Festpunkt liegt (siehe Katalog der Dienststellen).
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Dienststelle_Schluessel (data type)
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: 'Land' ist das Bundesland, das für die Bearbeitung des Festpunkts zuständig ist. Zur bundesweit eindeutigen Identifizierung eines Festpunktes gehören die Attributarten 'land' und 'punktkennung'.
Voidable:	false
Multiplicity:	1
Value type:	AX_Bundesland_Schluessel (data type)
Attribute:	
Name:	nameLagebeschreibung
Definition:	From the AAA 7.1 specification: 'NameLagebeschreibung' enthält den Namen bzw. eine Lagebeschreibung des Festpunktes.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	nutzerspezfischeBemerkung
Definition:	From the AAA 7.1 specification: 'Nutzerspezifische Bemerkungen' enthält Bemerkungen zu dem Festpunkt für den Nutzer.

Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString
Attribute:	
Name:	nutzungExtern
Definition:	From the AAA 7.1 specification: 'Nutzung Extern' gibt an, ob der Festpunkt für externe Nutzung zur Verwendung steht (true) bzw. die Verwendung auf den innerdienstlichen Bereich, der für den Festpunkt zuständigen Stelle, beschränkt ist (false).
Voidable:	false
Multiplicity:	1
Initial value:	true
Value type:	Boolean
Attribute:	
Name:	punktkennung
Definition:	From the AAA 7.1 specification: 'Punktkennung' ist ein Ordnungsmerkmal das in jedem Bundesland nach einer landesinternen Nummerierungsmethode vergeben und in den Metadaten erläutert wird. Zur bundesweit eindeutigen Identifizierung eines Festpunktes gehören die Attributarten 'Land' und 'Punktkennung'.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	punktvermarkung
Definition:	From the AAA 7.1 specification: 'Punktvermarkung' gibt an, mit welcher Marke der Festpunkt im Boden oder an baulichen Anlagen gekennzeichnet ist und auf welche Stelle der Punktvermarkung sich die Koordinaten, Höhen und Schwerewerte beziehen (siehe Katalog der AFIS-Vermarkungsarten). Wenn der Bezugspunkt in der Spalte 'Bezeichner' des Katalogs nicht anders definiert wird, ist es die höchste Stelle bzw. die Mitte der Oberfläche der Vermarkung.

Voidable: false

Multiplicity: 1

Value type: AX_Marke (enumeration)

Values

1000	**Marke, allgemein**
1100	**Stein**
1111	**Lochstein**
1112	**Vermessungspunktstein**
1120	**Unbehauener Feldstein**
1130	**Gemeinde- und Waldgrenzstein**
1131	**Gemeindegrenzstein**
1132	**Waldgrenzstein, Forstgrenzstein**
1140	**Kunstoffmarke**
1160	**Landesgrenzstein**
1190	**Stein mit Besonderheiten in Form oder Material**
1200	**Rohr**
1201	**Rohr mit Schutzkappe**
1202	**Rohr mit Kopf**
1203	**Rohr mit Bolzen, oberirdisch**
1210	**Eisenrohr**
1211	**Eisenrohr (mit Schutzkappe)**
1212	**Eisenrohr (ohne Schutzkappe)**
1220	**Kunststoffrohr**
1221	**Kunststoffrohr (mit Schutzkappe)**
1222	**Kunststoffrohr (ohne Schutzkappe)**
1230	**Drainrohr**
1240	**Rohr mit Schutzkasten**
1250	**Zementrohr**
1260	**Glasrohr**
1290	**Tonrohr**
1300	**Bolzen/Nagel**
1310	**Bolzen**

	1311	**Adapterbolzen**
	1320	**Nagel**
	1400	**Meisselzeichen (z.B. Kreuz, Kerbe, Anker)**
	1410	**Bohrloch**
	1500	**Pfahl**
	1600	**Sonstige Marke**
	1610	**Marke in Schutzberhälter**
	1620	**Flasche**
	1630	**Platte**
	1631	**Klinkerplatte**
	1632	**Granitplatte**
	1635	**Platte mit Loch**
	1640	**Hohlziegel**
	1650	**Klebemarke**
	1655	**Schlagmarke**
	1660	**Kanaldeckel (Kreuz des Gütesiegels auf Rand)**
	1670	**Marke besonderer Ausführung**
	1700	**Punkt dauerhaft und gut erkennbar festgelegt**
	1710	**Punkt der baulichen Anlage**
	1711	**Sockel (roh)**
	1712	**Sockel (verputzt)**
	1713	**Mauerecke (roh)**
	1714	**Mauerecke (verputzt)**
	1720	**Grenzsaeule**
	1800	**Pfeiler**
	1820	**Kegel**
	2100	**Festlegung 1. Ordnung, Kopf 30x30 cm, Bezugspunkt Platte**
	2101	**Festlegung 1. Ordnung, Bezugspunkt Kopf 30x30 cm**

2102	**Festlegung STN 1. Ordnung, Pfeilerkopf 30x30 cm, Bezugspunkt Platte 60x60 cm, Steinwürfel, Tonkegel**
2110	**Festlegung 2. bis 5. Ordnung, Kopf 16x16 oder 12x12 cm, Bezugspuknt Platte 30x30 cm**
2111	**Festlegung 2. bis 5. Ordnung, Bezugspunkt Kopf 16x16 oder 12x12 cm, Platte 30x30 cm**
2120	**Festlegung 2. bis 4. Ordnung, Kopf 20x20 cm, Bezugspunkt Platte**
2121	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 20x20 cm**
2130	**Festlegung 2. bis 4. Ordnung, Kopf 25x25 cm, Bezugspunkt Platte**
2131	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 25x25 cm**
2132	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Platte 30x30-40x40 cm**
2133	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Bolzen oder Rotgusskappe im Fels**
2134	**Festlegung TP-Feld, Pfeilerkopf 25x25 cm, Bezugspunkt Platte 35x35 cm**
2135	**Festlegung RBP-Feld, Pfeilerkopf 16x16 cm mit Gravur "TP" und "Dreieck", Bezugspunkt Platte 30x30-35x35 cm**
2140	**Plattformbolzen mit Aufschrift TP**
2150	**Turmbolzen mit Aufschrift TP**
2160	**Leuchtschraube oder -bolzen**
2161	**Schraube (vertikal)**
2162	**Messingmarke oder Messingbolzen**
2163	**Keramikbolzen, oberirdisch**
2164	**Bolzen im Fels, unterirdisch**
2165	**Rotgusskappe im Fels, unterirdisch**

	2166	**Messingbolzen (gewölbt), Aufschrift TP und Dreieck**
	2167	**3D-Messingbolzen (Durchmesser 5 cm) mit Inschrift RFP HESSEN**
	2170	**Turmbolzen, Festlegungsbolzen oder sonstiger Bolzen, keine weiteren Angaben bekannt oder gespeichert**
	2180	**Festlegung 2. Ordnung, Kopf 16x16 cm oder 12x12 cm, Bezugspunkt Platte 60x60 cm**
	2181	**Festlegung 2. Ordnung, Bezugspunkt Kopf 16x16 cm, oder 12x12 cm, Platte 60x60 cm**
	2190	**Festlegung 2. bis 3. Ordnung, Kopf 16x16 cm oder 12x12 cm, Platte 40x40 cm**
	2191	**Festlegung 2. bis 3. Ordnung, Bezugspunkt Kopf 16x16 cm, Platte 40x40 cm**
	2192	**Bergpfeiler ca. 60 cm lang, ohne Platte, Kopf 12x12 cm**
	2200	**Pfeiler mit Aufschrift AP**
	2201	**Pfeiler mit Aufschrift AP mit Platte, Bezugspunkt Kopf**
	2210	**Plattformbolzen mit der Aufschrift AP**
	2220	**Turmbolzen mit der Aufschrift AP**
	2230	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 1**
	2240	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 2**
	2250	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 3**
	2260	**Festlegung Sachsen-Anhalt SANREF (unterirdischer Granitpfeiler mit Kopfbolzen)**
	2261	**3D-Bolzen einzementiert in Kopfflaeche einer Festlegung STN 1. Ordnung**

2262	**3D-Bolzen in Kopfflaeche einer Festlegung TP-Feld**
2263	**3D-Bolzen in Kopffläche einer AF Sachsen 3. Ordnung**
2264	**3D-Bolzen in Kopfflaeche einer AF Sachsen 5. Ordnung**
2265	**3D-Bolzen in Granitplatte unter Schutzkasten oder Kunststoff-Schutzrohr**
2266	**3D-Bolzen in Kopffläche, einbetonierter Granitpfeiler 40x40x90 cm**
2267	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 25x25x100 cm**
2268	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 30x30x100 cm**
2269	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 50x50x100 cm**
2300	**Alte Festlegung der Kgl. Generalkomission und von Kurhessen, Rillenstein**
2310	**Alte Festlegung von Nassau, exz. ehemals 2.O.**
2320	**Alte Festlegung von Nassau, exz. ehemals 3.O.**
2330	**Alte Festlegung von Hessen-Darstadt, 3.O.**
2340	**Alte Festlegung von Hessen-Darmstadt, 4.O.**
2350	**Alte Festlegung von Hessen-Darmstadt, 3.O. exz.**
2360	**Alte Festlegung von Hessen-Darmstadt, 4.O. exz.**
2370	**Alte Festlegung von Westfalen, 2. O. exz.**
2400	**Alte Festlegung von Württemberg, exz.**
2410	**Alte Festlegung von Württemberg**

	2420	**Alte Festlegung von Mecklenburg, ehemals 1. bis 2. Ordnung**
	2430	**Alte Festlegung von Mecklenburg 2. bis 3. Ordnung**
	2440	**Alte Festlegung von Mecklenburg, ehemals 4. Ordnung**
	2450	**Alte Festlegung Sachsen, 1. und 2. Ordnung (Nagelsche Saeule)**
	2460	**Alte Festlegung Sachsen, 3. Ordnung, Pfeilerkopf 35x35 cm mit zentrischer Messingmarke**
	2470	**Alte Festlegung Sachsen, 5. Ordnung, Pfeilerkopf 25x25 cm mit zentrischer Mesingmarke**
	2500	**Alte Festlegung von Baden, exz., Typ 1**
	2510	**Alte Festlegung von Baden, exz., Typ 2**
	2540	**Alte Festlegung von Baden**
	2550	**Rohr mit Schutzkasten, Grundständer**
	2551	**Grundständer mit Messingmarke unter Schutzkasten**
	2552	**Marke (allgemein) unter Schutzkasten**
	2553	**Platte unter Schutzkasten**
	2560	**Rohr mit Verschlusskappe und aufgesetztem Lochstein**
	2600	**Alte Festlegung in Bayern, exz.**
	2601	**Gebohrter Granitstein (Bezusgpunkt) ueber Tonrohr**
	2602	**Gebohrter Granitstein ueber Tonrohr (Bezugspunkt)**
	2603	**Gebohrter Granitstein (Bezugspunkt) ueber Platte**
	2604	**Gebohrter Granitstein ueber Platte (Bezugspunkt)**
	2605	**Gebohrter Granitstein (Bezugspunkt) ueber Eisenrohr**

2606	**Gebohrter Granitstein ueber Eisenrohr (Bezugspunkt)**
2607	**Betonstein (Bezugspunkt) ueber Platte**
2608	**Betonstein ueber Platte (Bezugspunkt)**
2609	**Betonstein mit durchgehendem Bohrloch (Einschaltpunkt-Stein in Bayern)**
2610	**Alte Festlegung in Bayern**
2611	**KT-Stein der oesterreichischen Katastraltriangulation**
2612	**Stein (Bezugspunkt) ueber Platte**
2613	**Stein ueber Platte (Bezugspunkt)**
2614	**Stein (Bezugspunkt) ueber Tonrohr**
2615	**Stein ueber Tonrohr (Bezugspunkt)**
2616	**Stein mit Eisenrohr**
2620	**Alte Festlegung in Sachsen-Coburg, 3.O.**
2630	**Alte Festlegung in Sachsen-Coburg, 4.O.**
2640	**Alte Festlegung in Bayern un der Pfalz, HDNP-Stein**
2700	**Festlegung MP-Pfeiler**
2710	**Festlegung Orientierungspunkt, Bezugspunkt Platte**
2750	**Steinpfeiler**
2760	**Betonpfeiler**
2770	**Kreuz (gemeisselt)**
2800	**Knopf**
2810	**Mitte**
2820	**Spitze**
2830	**Kreuz (Mitte)**
2840	**Helmstange**
2850	**Fahnenstange**
2860	**Wetterstange**

2870	**Blitzableiter**
2880	**Antenne**
2890	**Rohrstange**
2900	**Platte, unterirdisch**
2901	**Platte einbetoniert mit zentrischem Messinbolzen**
2902	**Platte einbetoniert mit zentrischem Bohrloch**
2903	**Platte eibetoniert mit zentrischer Keramikmarke**
2904	**Platte eibetoniert mit zentrischer Messingmarke**
2910	**Steinwuerfel, unterirdisch**
2920	**Steinplatte, unterirdisch**
2930	**Platte, unterirdisch, 60x60 cm**
2940	**Platte, unterirdisch, 30x30 cm**
2950	**Platte, unterirdisch, mit Stehniet**
2951	**Platte, unterirdisch, mit Kopfbolzen**
2960	**Platte, unterirdisch, mit Schutzrohr**
2970	**Pfeiler, 30x40x90 cm, mit Stehniet**
2980	**Platte, mit Bolzen, unterirdisch, im Schacht**
3000	**Unterirdische Festlegung**
3011	**Unterirdische Festlegung mit Achatkugel**
3012	**Unterirdische Festlegung mit Halbkugel**
3013	**Unterirdische Festlegung mit Diabaeinsatz**
3014	**Unterirdische Festlegung im Schacht**
3015	**Kleine unterirdische Festlegung**
3020	**Unterirdischer Rammpfahl**
3030	**Unterirdischer Pfeilerbolzen**
3040	**Unterirdischer Bolzen**
3050	**Hamburger Flachpunkt**

3060	**Unterirdische Saeule**
3070	**Unterirdischer Rammstab**
3100	**Rohrfestpunkt**
3110	**Rohrfestpunkt, Hamburger Bauart**
3120	**Rohrfestpunkt, Oldenburger Bauart**
3130	**Rohrfestpunkt, Eider Bauart**
3140	**Rohrfestpunkt Nordrhein-Westfalen**
3150	**Rohrfestpunkt Nebenpunkt, flach gegründet**
3160	**Rohrfestpunkt, Celler Bauart**
3200	**Mauerbolzen**
3210	**Mauerbolzen, horizontale eingebracht**
3220	**Mauerbolzen, vertikal eingebracht**
3230	**Hoehenmarke**
3231	**Hoehenmarke (Koeniglich Preussische Landesaufnahme)**
3232	**Hoehenmarke (Koeniglich Saechsische Landesaufnahme)**
3233	**Hoehenmarke (Reichsamt fuer Landesaufnahme)**
3240	**Kugelbolzen**
3250	**Tonnenbolzen**
3260	**Landeshoehenbolzen**
3270	**Stehbolzen bzw. Bolzen vertikal**
3280	**Stehniete**
3290	**Sonstiger horizontaler Bolzen**
3300	**Pfeilerbolzen**
3301	**Pfeilerbolzen, Bezugspunkt Pfeileroberflaeche**
3310	**Pfeilerbolzen, Naturstein, Bolzen horizontal**
3311	**Pfeilerbolzen, Naturstein, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**

	3320	**Pfeilerbolzen, Naturstein, Bolzen vertikal**
	3330	**Pfeilerbolzen, Beton, Bolzen, horizontal**
	3331	**Pfeilerbolzen, Beton, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**
	3350	**Pfeilerniete, Naturstein, Niete vertikal**
	3400	**Rammpfahl**
	3410	**Rammpfahl, Bolzen horizontal**
	3420	**Rammpfahl, Bolzen vertikal**
	3810	**Schraubpfahl**
	3820	**Hektometerstein**
	3830	**Markstein**
	3840	**Schraubbolzen**
	3845	**Schraubeisen**
	3850	**Lochmarke/-bolzen (ohne Hoehentafel)**
	3860	**Lochmarke/-bolzen mit Hoehentafel**
	3870	**Festpunktstein**
	3880	**Eichpfahl**
	4100	**Gravimeterplatte 80x80 oder 60x60 cm**
	4110	**Gravimeternagel**
	4120	**Gravimeterpfeiler 20x20x100 cm**
	4130	**Gravimeterpfeiler 16x16x60 cm**
	4140	**Messingscheibe mit zentrischer Woelbung (Durchm. 8 cm)**
	4150	**Stehniete, Messing (Durchmesser 3cm, Aufschrift SFP)**
	4160	**Messplakette, Aufschrift**
	5100	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Betonpfeiler mt Fudament im festen Erdboden)**

5150	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Betonpfeiler mt Fudament im festen Erdboden)**
5200	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Gemauerter Pfeiler auf einem Bauwerk)**
5250	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Gemauerter Pfeiler auf einem Bauwerk)**
5300	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Stahlpfeiler auf einem Bauwerk)**
5350	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Stahlpfeiler auf einem Bauwerk)**
5400	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Seitlich befestigtes Stahlrohr am Bauwerk)**
5450	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Seitlich befestigtes Stahlrohr am Bauwerk)**
5500	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Antennentraeger)**
5550	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Antennentraeger)**
5900	**GNSS-Antennenhalterung, Oberflaeche Platte (Loch) (SL)**
9000	**Marke unter 'Bemerkung' naeher definiert**
9500	**Ohne Marke**
9600	**Abmarkung zeitweilig ausgesetzt**
9998	**Nach Quellenlage nicht zu spezifizieren**
9999	**Sonstiges**

Attribute:

Name: relativeHoehe

Definition: From the AAA 7.1 specification:

'Relative Höhe' gibt den Höhenunterschied in Meter an, um welchen der Höhenbezugspunkt der Vermarkung oberhalb

	(Vorzeichen '+') bzw. unterhalb (Vorzeichen '-') der Geländeoberfläche liegt.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	untergangsdatum
Definition:	From the AAA 7.1 specification: 'Untergangsdatum' gibt das Datum des dauerhaften Wegfalls der Vermarkung eines Punktes an.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	vermarkungsdatum
Definition:	From the AAA 7.1 specification: Mit dem 'Vermarkungsdatum' werden Zeitpunkte vermarkungsartrelevanter Änderungen am Festpunkt nachgewiesen. Dies sind - das Datum, an dem der Festpunkt erstmalig vermarkt wird, - Zeitpunkte, an denen sich die Vermarkungsart des Punktes durch Umvermarkung ändert, sowie - das Datum, an dem die Punktvermarkung dauerhaft entfällt.
Voidable:	false
Multiplicity:	0..1
Value type:	Date

1.9.12 AX_Flurstueck

AX_Flurstueck	
Definition:	From the AAA 7.1 specification: 'Flurstück' ist ein Teil der Erdoberfläche, der von einer im Liegenschaftskataster festgelegten Grenzlinie umschlossen und mit einer Nummer bezeichnet ist. Es ist die Buchungseinheit des Liegenschaftskatasters.
Subtype of:	CadastralParcel
Type:	Feature type

Attribute:

Name:	abweichenderRechtszustand
Definition:	From the AAA 7.1 specification: 'Abweichender Rechtszustand' ist ein Hinweis darauf, dass außerhalb des Grundbuches in einem durch Gesetz geregelten Verfahren der Bodenordnung (siehe Objektart 'Bau-, Raum- oder Bodenordnungsrecht', Attributart 'Art der Festlegung', Werte 1750, 1770, 2100 bis 2340, 2900) ein neuer Rechtszustand eingetreten ist und das amtliche Verzeichnis der jeweiligen ausführenden Stelle maßgebend ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean

Attribute:

Name:	amtlicheFlaeche
Definition:	From the AAA 7.1 specification: 'Amtliche Fläche' ist der im Liegenschaftskataster festgelegte Flächeninhalt des Flurstücks in Quadratmeter. Flurstücksflächen kleiner 0,5 Quadratmeter können mit bis zu zwei Nachkommastellen geführt werden, ansonsten ohne Nachkommastellen.
Voidable:	false
Multiplicity:	1
Value type:	Area

Attribute:

Name:	flurnummer
Definition:	From the AAA 7.1 specification: 'Flurnummer' ist die von der katasterführenden Stelle zur eindeutigen Bezeichnung vergebene Nummer einer Flur, die eine Gruppe von zusammenhängenden Flurstücken innerhalb einer Gemarkung umfasst.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

Attribute:

Name:	flurstuecksfolge
Definition:	From the AAA 7.1 specification:

	'Flurstücksfolge' ist eine weitere Angabe zur Flurstücksnummer zum Nachweis der Flurstücksentwicklung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	flurstueckskennzeichen
Definition:	From the AAA 7.1 specification: 'Flurstückskennzeichen' ist ein von der katasterführenden Stelle zur eindeutigen Bezeichnung des Flurstücks vergebenes Ordnungsmerkmal. Bildungsregel: Die Attributart setzt sich aus den nachfolgenden expliziten Attributarten in der angegebenen Reihenfolge zusammen: 1. Land (2 Stellen) 2. Gemarkungsnummer (4 Stellen) 3. Flurnummer (3 Stellen) 4. Flurstücksnummer 4.1 Zähler (5 Stellen) 4.2 Nenner (4 Stellen) 5. Flurstücksfolge (2 Stellen) Die Elemente sind rechtsbündig zu belegen, fehlende Stellen sind mit führenden Nullen zu belegen. Da die Flurnummer und die Flurstücksfolge optional sind, sind aufgrund der bundeseinheitlichen Definition im Flurstückskennzeichen die entsprechenden Stellen, sofern sie nicht belegt sind, durch Unterstrich "_" ersetzt. Gleiches gilt für Flurstücksnummern ohne Nenner, hier ist der fehlende Nenner im Flurstückskennzeichen durch Unterstriche zu ersetzen. Die Gesamtlänge des Flurstückkennzeichens beträgt immer 20 Zeichen.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	flurstuecksnummer

Definition:	From the AAA 7.1 specification: 'Flurstücksnummer' ist die Bezeichnung (Zähler/Nenner), mit der ein Flurstück innerhalb einer Flur (Flurnummer muss im Land vorhanden sein) oder Gemarkung identifiziert werden kann. Das Attribut setzt sich zusammen aus: 1. Spalte: Zähler 2. Spalte: Nenner Die 2. Spalte ist optional.
Voidable:	false
Multiplicity:	1
Value type:	AX_Flurstuecksnummer (data type)
Association role	
Name:	gehoertAnteiligZu
Voidable:	false
Multiplicity:	0..*
Value type:	AX_Flurstueck (feature type)
Attribute:	
Name:	gemarkung
Definition:	From the AAA 7.1 specification: 'Gemarkung' enthält die Eigenschaften aus dem Datentyp 'AX_Gemarkung_Schluessel': 'land' und 'gemarkungsnummer'.
Voidable:	false
Multiplicity:	1
Value type:	AX_Gemarkung_Schluessel (data type)
Attribute:	
Name:	gemeindezugehoerigkeit
Definition:	From the AAA 7.1 specification: 'Gemeindezugehörigkeit' enthält das Gemeindekennzeichen zur Zuordnung der Flurstücksdaten zu einer Gemeinde.
Voidable:	false
Multiplicity:	1
Value type:	AX_Gemeindekennzeichen (data type)
Attribute:	

Name:	lagebezeichnungOhneHausnummer
Definition:	From the AAA 7.1 specification: 'Lagebezeichnung ohne Hausnummer' ist die ortsübliche oder amtlich festgesetzte Benennung der Lage von Flurstücken und Gebäuden, die keine Hausnummer haben (z.B. Namen und Bezeichnungen von Gewannen, Straßen, Gewässern).
Voidable:	false
Multiplicity:	0..*
Value type:	AX_LagebezeichnungOhneHausnummer (data type)
Attribute:	
Name:	lagebezeichungMitHausnummer
Definition:	From the AAA 7.1 specification: 'Lagebezeichnung mit Hausnummer' ist die ortsübliche oder amtlich festgesetzte Benennung der Lage von Flurstücken und Gebäuden, die eine Lagebezeichnung mit Hausnummer haben. Hinweis zur Ableitung einer punktförmigen Geometrie zur Verortung der Hausnummer: Bei einer abweichenden Positionierung von der Standardposition liegt ein Präsentationsobjekt (Text) vor aus dem diese abgeleitet werden kann.
Voidable:	false
Multiplicity:	0..*
Value type:	AX_LagebezeichnungMitHausnummer (data type)
Attribute:	
Name:	rechtsbehelfsverfahren
Definition:	From the AAA 7.1 specification: 'Rechtsbehelfsverfahren' ist der Hinweis darauf, dass bei dem Flurstück ein laufendes Rechtsbehelfsverfahren anhängig ist.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean
Attribute:	
Name:	sonstigeEigenschaften
Definition:	From the AAA 7.1 specification:

	'Sonstige Eigenschaften' sind flurstücksbezogene Informationen, die in dem Datentyp AX_SonstigeEigenschaften enthalten sind. Die Attributart kommt vor, wenn sie übergangsweise im Rahmen der Migration aus bestehenden Verfahrenslösungen benötigt wird oder wenn die Angaben nicht als eigenständige raumbezogene Elementarobjekte aus dem Objektartenbereich 'Gesetzliche Festlegungen, Zuständigkeiten und Gebietseinheiten' geführt werden.
Voidable:	false
Multiplicity:	0..*
Value type:	AX_SonstigeEigenschaften_Flurstueck (data type)
Attribute:	
Name:	zustaendigeStelle
Definition:	From the AAA 7.1 specification: 'Flurstück' wird verwaltet von 'Dienststelle'. Diese Attributart wird nur dann belegt, wenn eine fachliche Zuständigkeit über eine Gemarkung bzw. Gemarkungsteil/Flur nicht abgebildet werden kann. Die Attributart enthält den Dienststellenschlüssel der Stelle, die fachlich für ein Flurstück zuständig ist.
Voidable:	false
Multiplicity:	0..*
Value type:	AX_Dienststelle_Schluessel (data type)
Attribute:	
Name:	zweifelhafterFlurstuecksnachweis
Definition:	From the AAA 7.1 specification: 'Zweifelhafter Flurstücksnachweis' ist eine Kennzeichnung eines Flurstücks, dessen Angaben nicht zweifelsfrei berichtigt werden können.
Voidable:	false
Multiplicity:	0..1
Value type:	Boolean

1.9.13 AX_Flurstuecksnummer

AX_Flurstuecksnummer	
Definition:	From the AAA 7.1 specification:

	'AX_Flurstücksnummer' ist ein Datentyp, der alle Eigenschaften für den Aufbau der Attributart 'Flurstücksnummer' enthält.
Type:	Data type
Attribute:	
Name:	nenner
Definition:	From the AAA 7.1 specification: Dieses Attribut enthält den Nenner der Flurstücknummer ohne führende Nullen. Diese sind gegebenenfalls bei der Erzeugung des Flurstückskennzeichens zu ergänzen.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zaehler
Definition:	From the AAA 7.1 specification: Dieses Attribut enthält den Zähler der Flurstücknummer ohne führende Nullen. Diese sind gebenefalls bei der Erzeugung des Flurstückskennzeichens zu ergänzen.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.14 AX_Gemarkung

AX_Gemarkung	
Definition:	From the AAA 7.1 specification: 'Gemarkung' ist ein Katasterbezirk, der eine zusammenhängende Gruppe von Flurstücken umfasst. Er kann von Gemarkungsteilen/Fluren unterteilt werden.
Subtype of:	CadastralZoning
Type:	Feature type
Attribute:	
Name:	gemeindezugehoerigkeit
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Gemeindekennzeichen (data type)

Attribute:	
Name:	istAmtsbezirkVon
Voidable:	false
Multiplicity:	0..*
Value type:	AX_Dienststelle_Schluessel (data type)
Attribute:	
Name:	schluessel
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Gemarkung_Schluessel (data type)

1.9.15 AX_Gemarkung_Schluessel

AX_Gemarkung_Schluessel	
Definition:	From the AAA 7.1 specification: Amtliche Verschlüsselung der Gemarkung.
Type:	Data type
Attribute:	
Name:	gemarkungsnummer
Definition:	From the AAA 7.1 specification: 'Gemarkungsnummer' enthält die von der katasterführenden Stelle zur eindeutigen Bezeichnung der Gemarkung vergebene Nummer innerhalb eines Bundeslandes.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: 'Land' enthält den Schlüssel für das Bundesland.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.16 AX_Gemeindekennzeichen

AX_Gemeindekennzeichen

Definition:	From the AAA 7.1 specification: 'Gemeindekennzeichen' ist die vom Statistischen Bundesamt veröffentlichte Schlüsselnummer des kommunalen Gebietes (Stadt-, Landgemeinde, gemeindefreies Gebiet). Das Gemeindekennzeichen (siehe Katalog der Gemeinden) besteht aus den Verschlüsselungen für : 1. Spalte: Land 2. Spalte: Regierungsbezirk 3. Spalte: Kreis (kreisfreie Stadt) 4. Spalte: Gemeinde und optional (siehe Katalog der Gemeindeteile) dem 5. Spalte: Gemeindeteil
Type:	Data type

Attribute:

Name:	gemeinde
Definition:	From the AAA 7.1 specification: Gemeinde.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

Attribute:

Name:	gemeindeteil
Definition:	From the AAA 7.1 specification: Gemeindeteil.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	kreis
Definition:	From the AAA 7.1 specification: Kreis.
Voidable:	false

Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	land
Definition:	From the AAA 7.1 specification: Land.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	regierungsbezirk
Definition:	From the AAA 7.1 specification: Regierungsbezirk. Diese Attributart ist optional, da nicht in allen Ländern Regierungsbezirke vorkommen.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.17 AX_Grenzpunkt

AX_Grenzpunkt	
Definition:	From the AAA 7.1 specification: 'Grenzpunkt' ist ein den Grenzverlauf bestimmender, meist durch Grenzzeichen gekennzeichneter Punkt.
Subtype of:	LA_Point
Type:	Feature type
Attribute:	
Name:	abmarkung
Definition:	From the AAA 7.1 specification: 'Abmarkung (Marke)' ist die Marke zur dauerhaften Kennzeichnung von Grenzpunkten im Boden und an baulichen Anlagen. Die Attributart ist hierarchisch in vier Stufen gegliedert. Die Gliederungsstufen ergeben sich aus den Werten für die Bezeichner (Tausender-, Hunderter-, Zehner- und Einerstelle).

Voidable: false

Multiplicity: 1

Value type: AX_Marke (enumeration)

Values

1000	**Marke, allgemein**
1100	**Stein**
1111	**Lochstein**
1112	**Vermessungspunktstein**
1120	**Unbehauener Feldstein**
1130	**Gemeinde- und Waldgrenzstein**
1131	**Gemeindegrenzstein**
1132	**Waldgrenzstein, Forstgrenzstein**
1140	**Kunstoffmarke**
1160	**Landesgrenzstein**
1190	**Stein mit Besonderheiten in Form oder Material**
1200	**Rohr**
1201	**Rohr mit Schutzkappe**
1202	**Rohr mit Kopf**
1203	**Rohr mit Bolzen, oberirdisch**
1210	**Eisenrohr**
1211	**Eisenrohr (mit Schutzkappe)**
1212	**Eisenrohr (ohne Schutzkappe)**
1220	**Kunststoffrohr**
1221	**Kunststoffrohr (mit Schutzkappe)**
1222	**Kunststoffrohr (ohne Schutzkappe)**
1230	**Drainrohr**
1240	**Rohr mit Schutzkasten**
1250	**Zementrohr**
1260	**Glasrohr**
1290	**Tonrohr**
1300	**Bolzen/Nagel**
1310	**Bolzen**

	1311	**Adapterbolzen**
	1320	**Nagel**
	1400	**Meisselzeichen (z.B. Kreuz, Kerbe, Anker)**
	1410	**Bohrloch**
	1500	**Pfahl**
	1600	**Sonstige Marke**
	1610	**Marke in Schutzberhälter**
	1620	**Flasche**
	1630	**Platte**
	1631	**Klinkerplatte**
	1632	**Granitplatte**
	1635	**Platte mit Loch**
	1640	**Hohlziegel**
	1650	**Klebemarke**
	1655	**Schlagmarke**
	1660	**Kanaldeckel (Kreuz des Gütesiegels auf Rand)**
	1670	**Marke besonderer Ausführung**
	1700	**Punkt dauerhaft und gut erkennbar festgelegt**
	1710	**Punkt der baulichen Anlage**
	1711	**Sockel (roh)**
	1712	**Sockel (verputzt)**
	1713	**Mauerecke (roh)**
	1714	**Mauerecke (verputzt)**
	1720	**Grenzsaeule**
	1800	**Pfeiler**
	1820	**Kegel**
	2100	**Festlegung 1. Ordnung, Kopf 30x30 cm, Bezugspunkt Platte**
	2101	**Festlegung 1. Ordnung, Bezugspunkt Kopf 30x30 cm**

2102	**Festlegung STN 1. Ordnung, Pfeilerkopf 30x30 cm, Bezugspunkt Platte 60x60 cm, Steinwürfel, Tonkegel**
2110	**Festlegung 2. bis 5. Ordnung, Kopf 16x16 oder 12x12 cm, Bezugspuknt Platte 30x30 cm**
2111	**Festlegung 2. bis 5. Ordnung, Bezugspunkt Kopf 16x16 oder 12x12 cm, Platte 30x30 cm**
2120	**Festlegung 2. bis 4. Ordnung, Kopf 20x20 cm, Bezugspunkt Platte**
2121	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 20x20 cm**
2130	**Festlegung 2. bis 4. Ordnung, Kopf 25x25 cm, Bezugspunkt Platte**
2131	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 25x25 cm**
2132	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Platte 30x30-40x40 cm**
2133	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Bolzen oder Rotgusskappe im Fels**
2134	**Festlegung TP-Feld, Pfeilerkopf 25x25 cm, Bezugspunkt Platte 35x35 cm**
2135	**Festlegung RBP-Feld, Pfeilerkopf 16x16 cm mit Gravur "TP" und "Dreieck", Bezugspunkt Platte 30x30-35x35 cm**
2140	**Plattformbolzen mit Aufschrift TP**
2150	**Turmbolzen mit Aufschrift TP**
2160	**Leuchtschraube oder -bolzen**
2161	**Schraube (vertikal)**
2162	**Messingmarke oder Messingbolzen**
2163	**Keramikbolzen, oberirdisch**
2164	**Bolzen im Fels, unterirdisch**
2165	**Rotgusskappe im Fels, unterirdisch**

	2166	**Messingbolzen (gewölbt), Aufschrift TP und Dreieck**
	2167	**3D-Messingbolzen (Durchmesser 5 cm) mit Inschrift RFP HESSEN**
	2170	**Turmbolzen, Festlegungsbolzen oder sonstiger Bolzen, keine weiteren Angaben bekannt oder gespeichert**
	2180	**Festlegung 2. Ordnung, Kopf 16x16 cm oder 12x12 cm, Bezugspunkt Platte 60x60 cm**
	2181	**Festlegung 2. Ordnung, Bezugspunkt Kopf 16x16 cm, oder 12x12 cm, Platte 60x60 cm**
	2190	**Festlegung 2. bis 3. Ordnung, Kopf 16x16 cm oder 12x12 cm, Platte 40x40 cm**
	2191	**Festlegung 2. bis 3. Ordnung, Bezugspunkt Kopf 16x16 cm, Platte 40x40 cm**
	2192	**Bergpfeiler ca. 60 cm lang, ohne Platte, Kopf 12x12 cm**
	2200	**Pfeiler mit Aufschrift AP**
	2201	**Pfeiler mit Aufschrift AP mit Platte, Bezugspunkt Kopf**
	2210	**Plattformbolzen mit der Aufschrift AP**
	2220	**Turmbolzen mit der Aufschrift AP**
	2230	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 1**
	2240	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 2**
	2250	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 3**
	2260	**Festlegung Sachsen-Anhalt SANREF (unterirdischer Granitpfeiler mit Kopfbolzen)**
	2261	**3D-Bolzen einzementiert in Kopfflaeche einer Festlegung STN 1. Ordnung**

2262	**3D-Bolzen in Kopfflaeche einer Festlegung TP-Feld**
2263	**3D-Bolzen in Kopffläche einer AF Sachsen 3. Ordnung**
2264	**3D-Bolzen in Kopfflaeche einer AF Sachsen 5. Ordnung**
2265	**3D-Bolzen in Granitplatte unter Schutzkasten oder Kunststoff-Schutzrohr**
2266	**3D-Bolzen in Kopffläche, einbetonierter Granitpfeiler 40x40x90 cm**
2267	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 25x25x100 cm**
2268	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 30x30x100 cm**
2269	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 50x50x100 cm**
2300	**Alte Festlegung der Kgl. Generalkomission und von Kurhessen, Rillenstein**
2310	**Alte Festlegung von Nassau, exz. ehemals 2.O.**
2320	**Alte Festlegung von Nassau, exz. ehemals 3.O.**
2330	**Alte Festlegung von Hessen-Darstadt, 3.O.**
2340	**Alte Festlegung von Hessen-Darmstadt, 4.O.**
2350	**Alte Festlegung von Hessen-Darmstadt, 3.O. exz.**
2360	**Alte Festlegung von Hessen-Darmstadt, 4.O. exz.**
2370	**Alte Festlegung von Westfalen, 2. O. exz.**
2400	**Alte Festlegung von Württemberg, exz.**
2410	**Alte Festlegung von Württemberg**

	2420	**Alte Festlegung von Mecklenburg, ehemals 1. bis 2. Ordnung**
	2430	**Alte Festlegung von Mecklenburg 2. bis 3. Ordnung**
	2440	**Alte Festlegung von Mecklenburg, ehemals 4. Ordnung**
	2450	**Alte Festlegung Sachsen, 1. und 2. Ordnung (Nagelsche Saeule)**
	2460	**Alte Festlegung Sachsen, 3. Ordnung, Pfeilerkopf 35x35 cm mit zentrischer Messingmarke**
	2470	**Alte Festlegung Sachsen, 5. Ordnung, Pfeilerkopf 25x25 cm mit zentrischer Mesingmarke**
	2500	**Alte Festlegung von Baden, exz., Typ 1**
	2510	**Alte Festlegung von Baden, exz., Typ 2**
	2540	**Alte Festlegung von Baden**
	2550	**Rohr mit Schutzkasten, Grundständer**
	2551	**Grundständer mit Messingmarke unter Schutzkasten**
	2552	**Marke (allgemein) unter Schutzkasten**
	2553	**Platte unter Schutzkasten**
	2560	**Rohr mit Verschlusskappe und aufgesetztem Lochstein**
	2600	**Alte Festlegung in Bayern, exz.**
	2601	**Gebohrter Granitstein (Bezusgpunkt) ueber Tonrohr**
	2602	**Gebohrter Granitstein ueber Tonrohr (Bezugspunkt)**
	2603	**Gebohrter Granitstein (Bezugspunkt) ueber Platte**
	2604	**Gebohrter Granitstein ueber Platte (Bezugspunkt)**
	2605	**Gebohrter Granitstein (Bezugspunkt) ueber Eisenrohr**

2606	**Gebohrter Granitstein ueber Eisenrohr (Bezugspunkt)**
2607	**Betonstein (Bezugspunkt) ueber Platte**
2608	**Betonstein ueber Platte (Bezugspunkt)**
2609	**Betonstein mit durchgehendem Bohrloch (Einschaltpunkt-Stein in Bayern)**
2610	**Alte Festlegung in Bayern**
2611	**KT-Stein der oesterreichischen Katastraltriangulation**
2612	**Stein (Bezugspunkt) ueber Platte**
2613	**Stein ueber Platte (Bezugspunkt)**
2614	**Stein (Bezugspunkt) ueber Tonrohr**
2615	**Stein ueber Tonrohr (Bezugspunkt)**
2616	**Stein mit Eisenrohr**
2620	**Alte Festlegung in Sachsen-Coburg, 3.O.**
2630	**Alte Festlegung in Sachsen-Coburg, 4.O.**
2640	**Alte Festlegung in Bayern un der Pfalz, HDNP-Stein**
2700	**Festlegung MP-Pfeiler**
2710	**Festlegung Orientierungspunkt, Bezugspunkt Platte**
2750	**Steinpfeiler**
2760	**Betonpfeiler**
2770	**Kreuz (gemeisselt)**
2800	**Knopf**
2810	**Mitte**
2820	**Spitze**
2830	**Kreuz (Mitte)**
2840	**Helmstange**
2850	**Fahnenstange**
2860	**Wetterstange**

2870	**Blitzableiter**
2880	**Antenne**
2890	**Rohrstange**
2900	**Platte, unterirdisch**
2901	**Platte einbetoniert mit zentrischem Messinbolzen**
2902	**Platte einbetoniert mit zentrischem Bohrloch**
2903	**Platte eibetoniert mit zentrischer Keramikmarke**
2904	**Platte eibetoniert mit zentrischer Messingmarke**
2910	**Steinwuerfel, unterirdisch**
2920	**Steinplatte, unterirdisch**
2930	**Platte, unterirdisch, 60x60 cm**
2940	**Platte, unterirdisch, 30x30 cm**
2950	**Platte, unterirdisch, mit Stehniet**
2951	**Platte, unterirdisch, mit Kopfbolzen**
2960	**Platte, unterirdisch, mit Schutzrohr**
2970	**Pfeiler, 30x40x90 cm, mit Stehniet**
2980	**Platte, mit Bolzen, unterirdisch, im Schacht**
3000	**Unterirdische Festlegung**
3011	**Unterirdische Festlegung mit Achatkugel**
3012	**Unterirdische Festlegung mit Halbkugel**
3013	**Unterirdische Festlegung mit Diabaeinsatz**
3014	**Unterirdische Festlegung im Schacht**
3015	**Kleine unterirdische Festlegung**
3020	**Unterirdischer Rammpfahl**
3030	**Unterirdischer Pfeilerbolzen**
3040	**Unterirdischer Bolzen**
3050	**Hamburger Flachpunkt**

3060	**Unterirdische Saeule**
3070	**Unterirdischer Rammstab**
3100	**Rohrfestpunkt**
3110	**Rohrfestpunkt, Hamburger Bauart**
3120	**Rohrfestpunkt, Oldenburger Bauart**
3130	**Rohrfestpunkt, Eider Bauart**
3140	**Rohrfestpunkt Nordrhein-Westfalen**
3150	**Rohrfestpunkt Nebenpunkt, flach gegründet**
3160	**Rohrfestpunkt, Celler Bauart**
3200	**Mauerbolzen**
3210	**Mauerbolzen, horizontale eingebracht**
3220	**Mauerbolzen, vertikal eingebracht**
3230	**Hoehenmarke**
3231	**Hoehenmarke (Koeniglich Preussische Landesaufnahme)**
3232	**Hoehenmarke (Koeniglich Saechsische Landesaufnahme)**
3233	**Hoehenmarke (Reichsamt fuer Landesaufnahme)**
3240	**Kugelbolzen**
3250	**Tonnenbolzen**
3260	**Landeshoehenbolzen**
3270	**Stehbolzen bzw. Bolzen vertikal**
3280	**Stehniete**
3290	**Sonstiger horizontaler Bolzen**
3300	**Pfeilerbolzen**
3301	**Pfeilerbolzen, Bezugspunkt Pfeileroberflaeche**
3310	**Pfeilerbolzen, Naturstein, Bolzen horizontal**
3311	**Pfeilerbolzen, Naturstein, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**

	3320	**Pfeilerbolzen, Naturstein, Bolzen vertikal**
	3330	**Pfeilerbolzen, Beton, Bolzen, horizontal**
	3331	**Pfeilerbolzen, Beton, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**
	3350	**Pfeilerniete, Naturstein, Niete vertikal**
	3400	**Rammpfahl**
	3410	**Rammpfahl, Bolzen horizontal**
	3420	**Rammpfahl, Bolzen vertikal**
	3810	**Schraubpfahl**
	3820	**Hektometerstein**
	3830	**Markstein**
	3840	**Schraubbolzen**
	3845	**Schraubeisen**
	3850	**Lochmarke/-bolzen (ohne Hoehentafel)**
	3860	**Lochmarke/-bolzen mit Hoehentafel**
	3870	**Festpunktstein**
	3880	**Eichpfahl**
	4100	**Gravimeterplatte 80x80 oder 60x60 cm**
	4110	**Gravimeternagel**
	4120	**Gravimeterpfeiler 20x20x100 cm**
	4130	**Gravimeterpfeiler 16x16x60 cm**
	4140	**Messingscheibe mit zentrischer Woelbung (Durchm. 8 cm)**
	4150	**Stehniete, Messing (Durchmesser 3cm, Aufschrift SFP)**
	4160	**Messplakette, Aufschrift**
	5100	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Betonpfeiler mt Fudament im festen Erdboden)**

5150	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Betonpfeiler mt Fudament im festen Erdboden)**
5200	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Gemauerter Pfeiler auf einem Bauwerk)**
5250	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Gemauerter Pfeiler auf einem Bauwerk)**
5300	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Stahlpfeiler auf einem Bauwerk)**
5350	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Stahlpfeiler auf einem Bauwerk)**
5400	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Seitlich befestigtes Stahlrohr am Bauwerk)**
5450	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Seitlich befestigtes Stahlrohr am Bauwerk)**
5500	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Antennentraeger)**
5550	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Antennentraeger)**
5900	**GNSS-Antennenhalterung, Oberflaeche Platte (Loch) (SL)**
9000	**Marke unter 'Bemerkung' naeher definiert**
9500	**Ohne Marke**
9600	**Abmarkung zeitweilig ausgesetzt**
9998	**Nach Quellenlage nicht zu spezifizieren**
9999	**Sonstiges**

Attribute:

Name: ausgesetzteAbmarkung

Definition: From the AAA 7.1 specification:

'Ausgesetzte Abmarkung' ist eine Kennzeichnung der Stelle, die die Abmarkung eines Grenzpunktes zeitweilig ausgesetzt bzw. zurückgestellt hat (siehe Katalog der Dienststellen).

Voidable: false

Multiplicity: 0..1

Value type: AX_Dienststelle_Schluessel (data type)

Attribute:

Name: bemerkungZurAbmarkung

Definition: From the AAA 7.1 specification:

'Bemerkung zur Abmarkung' ist eine Angabe zur Sicherung der Abmarkung.

Voidable: false

Multiplicity: 0..1

Value type: AX_BemerkungZurAbmarkung_Grenzpunkt (enumeration)

Values

1000	**Abmarkung unterirdisch gesichert**
2000	**Abmarkung exzentrisch gesichert**
3000	**Abmarkung uterirdisch und exzentrisch gesichert**
4000	**Ohne unterirdische oder exzentrische Sicherung**

Attribute:

Name: besonderePunktnummer

Definition: From the AAA 7.1 specification:

'Besondere Punktnummer' ist eine durch amtliche Stellen vergebene fachspezifische Kennung für einen Grenzpunkt (z.B.: Landes- oder Bundesgrenzpunktes).

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: festgestellterGrenzpunkt

Definition: From the AAA 7.1 specification:

'Festgestellter Grenzpunkt' ist ein Hinweis darauf, dass der Grenzpunkt Bestandskraft erlangt hat.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: gruendeDerAusgesetztenAbmarkung

Definition: From the AAA 7.1 specification:

'Gründe der ausgesetzten Abmarkung' zeigt die Gründe auf, weshalb eine Abmarkung zeitweilig ausgesetzt ist.

Voidable: false

Multiplicity: 0..1

Value type: AX_GruendeDerAusgesetztenAbmarkung_Grenzpunkt (enumeration)

Values

1000	**Grenzpunkt durch bauliche Anlage ausreichend gekennzeichnet**
2000	**Grenzpunkt liegt innerhalb einer baulichen Anlage**
3000	**Grenzpunkt in oeffentlich-rechtlichem Bodenordungsverfahren**
4000	**Grenzpunkt liegt innerhalb eines Baugebietes**
5000	**Grenzpunkt liegt innerhalb oder an einem Gewaesser**
6000	**Keine Abmarkung aufgrund von anderweitigen Hindernissen**
7000	**Abmarkung wuerde unzumutbare Schaeden verursachen**
8000	**Langrfistige gemeinschaftliche Nutzung der angrenzenden Flurstuecke**
9000	**Angrenzende Flurstuecke dienen dem Gemeingebrauch**
9999	**Sonstiges**

Attribute:

Name: horizontfreiheit

Definition: From the AAA 7.1 specification:

'Horizontfreiheit' beschreibt die Abschattung bei Satellitenmessverfahren.

Voidable: false

Multiplicity: 0..1

Value type: AX_Horizontfreiheit_Grenzpunkt (enumeration)

Values

1000	**Uneingeschraenkt**
2000	**Eingeschraenkt**
3000	**Nicht geeignet**

Attribute:

Name: punktkennung

Definition: From the AAA 7.1 specification:

'Punktkennung' ist ein von der katasterführenden Stelle vergebenes Ordnungsmerkmal.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: relativeHoehe

Definition: From the AAA 7.1 specification:

'Relative Höhe' ist die Angabe der Höhe in Meter der 'Abmarkung (Marke)' oberhalb der Erdoberfläche oder der Tiefe in Meter unterhalb der Erdoberfläche.

(Vorzeichenregel: oberhalb der Erdoberfläche '+', unterhalb der Erdoberfläche '-'.)

Voidable: false

Multiplicity: 0..1

Value type: Length

Attribute:

Name: sonstigeEigenschaften

Definition: From the AAA 7.1 specification:

'Sonstige Eigenschaft' sind Informationen zum Grenzpunkt.

Sonstige Eigenschaften werden im Rahmen der Migration aus bestehenden Verfahrenslösungen übernommen.

Voidable: false

Multiplicity: 0..*

Value type: CharacterString

Association role

Name:	zeigtAuf
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Grenzpunkt (feature type)
Attribute:	
Name:	zeitpunktDerEnstehung
Definition:	From the AAA 7.1 specification: 'Zeitpunkt der Entstehung' ist der Zeitpunkt oder das Entstehungsjahr, zu dem der Grenzpunkt fachlich entstanden ist. Das Attribut kommt vor, wenn der Zeitpunkt der Entstehung von dem Zeitpunkt abweicht, der systemseitig bei der Eintragung in den Bestandsdaten als Anfang der Lebenszeit (siehe Lebenszeitintervall bei Objekten) gesetzt wird. Die Regelungen hierzu sind länderspezifisch gefasst.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zustaendigeStelle
Definition:	From the AAA 7.1 specification: 'Zuständige Stelle' enthält den Dienststellenschlüssel der Stelle, die eine Zuständigkeit besitzt.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Dienststelle_Schluessel (data type)
Attribute:	
Name:	zwischenmarke
Definition:	From the AAA 7.1 specification: 'Zwischenmarke' ist eine gebietsweise übliche Bezeichnung für ein Grenzzeichen, das in eine geradlinige Flurstücksgrenze eingebracht ist, um den Grenzverlauf bei fehlender Sichtverbindung oder großer Entfernung zwischen den Grenzzeichen ausreichend erkennbar zu machen.
Voidable:	false
Multiplicity:	0..1

Value type:	Boolean

1.9.18 AX_Hoehenfestpunkt

AX_Hoehenfestpunkt

Definition:	From the AAA 7.1 specification: 'Höhenfestpunkt' (HFP) ist ein Festpunkt der Grundlagenvermessung für die Höhe. Die Eigenschaften 'Land' und 'Punktkennung' sind objektbildend. Das Lebenszeitintervall eines Objektes 'Höhenfestpunkt' beginnt mit der Vergabe und endet mit dem Untergang der Attributart 'Punktkennung'. Ein noch nicht untergegangenes Objekt der Objektart 'Höhenfestpunkt' muss mindestens ein REO 'PunktortAU' mit 2D- oder 3D-Koordinaten enthalten. HFP-Unterlagen, die außerhalb von AFIS geführt werden, sind in einer Fachdatenverbindung zu führen.
Subtype of:	AX_Festpunkt
Type:	Feature type

Attribute:

Name:	nivlinie
Definition:	From the AAA 7.1 specification: Bezeichnung der Zugehörigkeit eines Höhenfestpunktes zu einer NIV-Linie.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString

Attribute:

Name:	ordnung
Definition:	From the AAA 7.1 specification: 'Ordnung' gibt eine Klassifikation des HFP an.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Ordnung_Hoehenfestpunkt (enumeration)
Values	1000 — **1. Ordnung**

	1001	**1. Ordnung - Netzverdichtung GPS (Niedersachsen)**
	2000	**2. Ordnung**
	3000	**3. Ordnung**
	3001	**3. Ordnung - nivellitisch bestimmter Bodenpunkt fuer Referenzstation**
	4000	**4. Ordnung**
	6000	**UEH - Uebergeordneter Hoehenfestpunkt (Berlin)**
	9998	**Nach Quellenlage nicht spezifizierbar**

Attribute:

Name: qualitaetsangaben

Definition: From the AAA 7.1 specification:

Qualitätsangaben zu den Daten des Festpunkts.

Voidable: false

Multiplicity: 0..1

Value type: AX_DQHoehenfestpunkt (data type)

1.9.19 AX_Klassifikation_Lagefestpunkt

AX_Klassifikation_Lagefestpunkt

Definition: From the AAA 7.1 specification:

'Klassifikation' gibt Ordnung, Hierarchiestufe und ggf. Wertigkeit des Festpunkts an.

Type: Data type

Attribute:

Name: hierachiestufe3D

Definition: From the AAA 7.1 specification:

Hierarchiestufe des LFP.

Voidable: false

Multiplicity: 0..1

Value type: AX_Klassifikation_Hierachiestufe3D_Lagefestpunkt (enumeration)

Values

1000	**A**
2000	**B**

3000	**C**
4000	**D**
5000	**E**
9998	**Nach Quellenlage nicht zu spezifizieren**

Attribute:

Name: ordnung

Definition: From the AAA 7.1 specification:

Ordnung des LFP.

Voidable: false

Multiplicity: 0..1

Value type: AX_Klassifikation_Ordnung_Lagefestpunkt (enumeration)

Values

1000	**1. Ordnung**
2000	**2. Ordnung**
3000	**3. Ordnung**
4000	**4. Ordnung**
5000	**5. Ordnung**
6000	**UEL - Uebergeordneter Lagefestpunkt (Berlin)**
9998	**Nach Quellenlage nicht zu spezifizieren**
9999	**Sonstiges**

Attribute:

Name: wertigkeit

Definition: From the AAA 7.1 specification:

Wertigkeit des LFP.

Voidable: false

Multiplicity: 0..1

Value type: AX_Klassifiaktion_Wertigkeit_Lagefestpunkt (enumeration)

Values

1000	**Fundamentalpunkt**
2000	**Uebergeordneter Festpunkt**
3000	**Geodaetischer Grundnetzpunkt**
4000	**Gebrauchsfestpunkt**

	5000	**Untergeordneter Festpunkt**
	9998	**Nach Quellenlage nicht zu spezifizieren**
	9999	**Sonstiges**

1.9.20 AX_LagebezeichnungMitHausnummer

AX_LagebezeichnungMitHausnummer

Definition: From the AAA 7.1 specification:

'Lagebezeichnung mit Hausnummer' ist die ortsübliche oder amtlich festgesetzte Benennung der Lage von Flurstücken und Gebäuden, die eine Lagebezeichnung mit Hausnummer haben.

Hinweis zur Ableitung einer punktförmigen Geometrie zur Verortung der Hausnummer:

Bei einer abweichenden Positionierung von der Standardposition liegt ein Präsentationsobjekt (Text) vor aus dem diese abgeleitet werden kann.

Type: Data type

Attribute:

Name: hausnummer

Definition: From the AAA 7.1 specification:

'Hausnummer' ist die von der Gemeinde für ein bestehendes oder geplantes Gebäude vergebene Nummer und ggf. einem Adressierungszusatz. Diese Attributart wird in Verbindung mit dem Straßennamen (verschlüsselte oder unverschlüsselte Lagebezeichnung) vergeben.

Voidable: false

Multiplicity: 1

Value type: CharacterString

Attribute:

Name: ortsteil

Definition: From the AAA 7.1 specification:

'Ortsteil' ist eine Ergänzung zur Lagebezeichnung um den Ortsteil.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

1.9.21 AX_LagebezeichnungOhneHausnummer

AX_LagebezeichnungOhneHausnummer	
Definition:	From the AAA 7.1 specification: 'Lagebezeichnung ohne Hausnummer' ist die ortsübliche oder amtlich festgesetzte Benennung der Lage von Flurstücken und Gebäuden, die keine Hausnummer haben (z.B. Namen und Bezeichnungen von Gewannen, Straßen, Gewässern).
Type:	Data type
Attribute:	
Name:	ortsteil
Definition:	From the AAA 7.1 specification: 'Ortsteil' ist eine Ergänzung zur Lagebezeichnung um den Ortsteil.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	zusaetzlicheLagebezeichnung
Definition:	From the AAA 7.1 specification: 'Zusatz zur Lagebezeichnung' ist eine Ergänzung zur Lagebezeichnung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.9.22 AX_Lagefestpunkt

AX_Lagefestpunkt	
Definition:	From the AAA 7.1 specification: 'Lagefestpunkt' (LFP) ist ein Festpunkt der Grundlagenvermessung für die räumliche Position (3D) oder die Lage (2D).
Subtype of:	AX_Festpunkt
Type:	Feature type
Attribute:	

Name: funktion

Definition: From the AAA 7.1 specification:

'Funktion' gibt an, welche Stellung der Punkt in der TP-Punktgruppe hat

Voidable: false

Multiplicity: 0..1

Value type: AX_Funktion_Lagefestpunkt (enumeration)

Values

1000	**Zentrum**
2000	**Exzentrum**
3000	**Zwillingspunkt, Orientierungspunkt**
4000	**Versicherungspunkt**

Attribute:

Name: klassifikation

Definition: From the AAA 7.1 specification:

'Klassifikation' gibt Ordnung, Hierarchiestufe und ggf. Wertigkeit des LFP an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Klassifikation_Lagefestpunkt (data type)

Attribute:

Name: pfeilerhoehe

Definition: From the AAA 7.1 specification:

'Pfeilerhöhe' gibt bei Vermarkungen, die aus Pfeiler und Platte bestehen, die Höhendifferenz zwischen Pfeileroberfläche und Plattenoberfläche sowie das Messdatum an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Pfeilerhoehe_Lagefestpunkt (data type)

Attribute:

Name: qualitaetsangaben

Definition: From the AAA 7.1 specification:

Qualitätsangaben zu den Daten des Festpunkts.

Voidable: false

Multiplicity:	0..1
Value type:	AX_DQFestpunkt (data type)

1.9.23 AX_Nammensnummer

AX_Nammensnummer

Definition:	From the AAA 7.1 specification: 'Namensnummer' ist die laufende Nummer der Eintragung, unter welcher der Eigentümer oder Erbbauberechtigte im Buchungsblatt geführt wird. Rechtsgemeinschaften werden auch unter AX_Namensnummer geführt.
Subtype of:	AbstractFeatureWithLifespan
Type:	Feature type

Attribute:

Name:	anteil
Definition:	From the AAA 7.1 specification: 'Anteil' ist der Anteil der Berechtigten in Bruchteilen (Par. 47 GBO) an einem gemeinschaftlichen Eigentum (Grundstück oder Recht).
Voidable:	false
Multiplicity:	0..1
Value type:	Fraction (data type)

Attribute:

Name:	artDerRechtsgemeinschaft
Definition:	From the AAA 7.1 specification: 'Art der Rechtsgemeinschaft' ist die Art des für die Gesamthandgemeinschaft maßgebenden Rechtsverhältnisses.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_ArtDerRechtsgemeinschaft_Nammensnummer (enumeration)

Values

1000	**Erbengemeinschaft**
2000	**Gütergemeinschaft**
3000	**BGB-Gesellschaft**
4010	**Gesamtberechtigte gemaess §428 BGB**

	4020	**Gesamtberechtigte gemaess §432 BGB**
	4030	**Mitglieder eines nicht eingetragenen Vereins**
	4040	**Fortgesetzte Guetergemeinschaft**
	4050	**Beendete, nicht auseinandergesetzte Guetergemeinschaft**
	4060	**Errungenschaftsgemeinschaft**
	4070	**Fortgesetzte Errungenschaftsgemeinschaft**
	4080	**Beendete, nicht auseinandergesetzte Errungenschaftsgemeinschaft**
	4090	**Fahrnisgemeinschaft**
	4100	**Fortgesetzte Fahrnisgemeinschaft**
	4110	**Beendete, nicht auseinandergesetzte Fahrnisgemeinschaft**
	4120	**Eigentums- und Vermoegensgemeinschaft nach FGB**
	4130	**Beendete, nicht auseindandergesetzte Eigentums- und Vermoegensgemeinschaft nach FGB**
	4140	**Altrechtliche Gemeinschaft**
	9999	**Sonstiges**

Association role

Name: bennent

Voidable: false

Multiplicity: 0..1

Value type: AX_Person (feature type)

Attribute:

Name: beschriebDerRechtsgemeinschaft

Definition: From the AAA 7.1 specification:

'Beschrieb der Rechtsgemeinschaft' ist der Name oder die juristische Bezeichnung der Rechtsgemeinschaft.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: eigentuemerArt

Definition: From the AAA 7.1 specification:

'Eigentümerart' ist die Kategorie des Eigentums.

Die Information wird nach Einführung des Datenbankgrundbuches (DaBaG) von der Grundbuchverwaltung nicht mehr übermittelt.

Voidable: false

Multiplicity: 0..1

Value type: AX_Eigentuemerart_Nammensnummer (enumeration)

Values

1000	**Natuerliche Person**
1100	**Natuerliche Person - Alleingeigentum oder Ehepartner**
1200	**Natuerliche Person - Wohnsitz im Land**
1300	**Natuerliche Person - Wohnsitz ausserhalb des Landes**
1500	**Natuerliche Person - Gemeinschaftseigentum**
2000	**Juristische Person**
2100	**Gemeinnuetzige Bau-, Wohnungs-, oder Siedlungsgesellschaft oder -genossenschaft einschliesslich Heimstaette**
2200	**Sonstige gemeinnuetzige Institution (Traeger von Krankenhaeusern, Alternheimen usw.)**
2300	**Privates Wohnungsunternehmen, private Baugesellschaft u.ae.**
2400	**Kreditinstitut**
2500	**Versicherungsunternehmen**
2900	**Andere Unternehmen, Gesellschaften usw.**
3000	**Koerperschaften**
3100	**Stiftung**
4000	**Kirchliches Eigentum**

4100	**Evangelische Kirche**
4200	**Katholische Kirche**
4900	**Andere Kirchen, Religionsgemeinschaften usw.**
5100	**Bundesrepublik Deutschland**
5101	**Bundesrepublik Deutschland, Bundesstrassenverwaltung**
5102	**Bundesrepublik Deutschland, Bundeswehrverwaltung**
5103	**Bundesrepublik Deutschland, Forstverwaltung**
5104	**Bundesrepublik Deutschland, Finanzverwaltung**
5105	**Bundesrepublik Deutschland, Zivilschutz**
5106	**Bundesrepublik Deutschland, Wasserstrassenverwaltung**
5107	**Bundesrepublik Deutschland, Bundeseisenbahnvermögen**
5108	**Bundesanstalt fuer Immobilienaufgaben**
5210	**Eigentum des Volkes nach DDR-Recht**
5220	**Eigentum der Genossenschaften und deren Einrichtungen**
5230	**Eigentum der gesellschaftlichen Organisationen und deren Einrichtungen**
5240	**Kommunale Gebietskoerperschaften nach DDR-Recht**
5300	**Auslaendischer Staat**
5400	**Kreis**
5500	**Gemeinde**
5600	**Kommunale Gebietskoerperschaften**
5700	**Andere Gebietskoerperschaften, Regionalverbaende usw.**
5800	**Zweckverbaende, Kommunale Betriebe**

5920	**Eigenes Bundesland**
5921	**Eigenes Bundesland, Denkmalpflege**
5922	**Eigenes Bundesland, Domaenenverwaltung**
5923	**Eigenes Bundesland, Eichverwaltung**
5924	**Eigenes Bundesland, Finanzverwaltung**
5925	**Eigenes Bundesland, Forstverwaltung**
5926	**Eigenes Bundesland, Gesundheitswesen**
5927	**Eigenes Bundesland, Polizeiverwaltung**
5928	**Eigenes Bundesland, innere Verwaltung**
5929	**Eigenes Bundesland, Justizverwaltung**
5930	**Eigenes Bundesland, Kultusverwaltung**
5931	**Eigenes Bundesland, Landespflanzenschutzverwaltung**
5932	**Eigenes Bundesland, Arbeitsverwaltung**
5933	**Eigenes Bundesland, Sozialwesen**
5934	**Eigenes Bundesland, Landesbetrieb Strassen und Verkehr**
5935	**Eigenes Bundesland, Umweltverwaltung**
5936	**Eigenes Bundesland, Vermessungs- und Katasterverwaltung**
5937	**Eigenes Bundesland, Wasserwirtschaftsverwaltung**
5938	**Eigenes Bundesland, Wirtschaftsverwaltung**
5939	**Eigenes Bundesland, Liegenschafts- und Baubetreuung**
5940	**Eigenes Bundesland, Naturschutzverwaltung**

6000	**Anderes Bundesland (allg.)**
6001	**Schleswig-Holstein**
6002	**Hamburg**
6003	**Niedersachsen**
6004	**Bremen**
6005	**Nordrhein-Westfalen**
6006	**Hessen**
6007	**Rheinland-Pfalz**
6008	**Baden-Wuerttemberg**
6009	**Bayern**
6010	**Saarland**
6012	**Brandenburg**
6011	**Berlin**
6013	**Mecklenburg-Vorpommern**
6014	**Sachsen**
6015	**Sachsen-Anhalt**
6016	**Thueringen**
7100	**Deutsche Bahn AG**
8000	**Herrenlos**
9000	**Eigentuemer unbekannt**

Association role

Name: istBestandteilVon

Voidable: false

Multiplicity: 1

Value type: AX_Buchungsblatt (feature type)

Attribute:

Name: laufendeNummerNachDIN1421

Definition: From the AAA 7.1 specification:

'Laufende Nummer nach DIN 1421' ist die interne laufende Nummer für die Rangfolge der Person, die nach den Vorgaben aus DIN 1421 strukturiert ist.

Voidable: false

Multiplicity: 0..1

Value type:	CharacterString
Attribute:	
Name:	nummer
Definition:	From the AAA 7.1 specification: 'Nummer' ist die laufende Nummer der Eintragung gemäß Abteilung 1 Grundbuchblatt, unter der eine Person aufgeführt ist (z.B. 1 oder 1a).
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	strichblattnummer
Definition:	From the AAA 7.1 specification: 'Strichblattnummer' ist eine Unternummer der Grundbuchblattnummer. Sie wird der Attributart 'Nummer' als Präfix vorangestellt.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.9.24 AX_Netzpunkt

AX_Netzpunkt	
Definition:	From the AAA 7.1 specification:
Subtype of:	LA_Point
Supertype of:	AX_Aufnahmepunkt AX_Sicherungspunkt AX_SonstigerVermessungspunkt
Type:	Feature type
Attribute:	
Name:	horizontfreiheit
Definition:	From the AAA 7.1 specification: 'Horizontfreiheit' beschreibt die Abschattung bei Satellitenmessverfahren.
Voidable:	false

Multiplicity: 0..1

Value type: AX_Horizontfreiheit_Netzpunkt (enumeration)

Values

1000	**Uneingeschraenkt**
2000	**Eingesschraenkt**
3000	**Nicht geeignet**

Attribute:

Name: punktkennung

Definition: From the AAA 7.1 specification:

'Punktkennung' ist ein von der katasterführenden Stelle vergebenes Ordnungsmerkmal.

Voidable: false

Multiplicity: 0..1

Value type: CharacterString

Attribute:

Name: relativeHoehe

Definition: From the AAA 7.1 specification:

'Relative Höhe' ist die Angabe der Höhe in Meter der 'Vermarkung (Marke)' oberhalb der Erdoberfläche oder der Tiefe in Meter unterhalb der Erdoberfläche.

(Vorzeichenregel: oberhalb der Erdoberfläche '+', unterhalb der Erdoberfläche '-'.)

Voidable: false

Multiplicity: 0..1

Value type: Length

Attribute:

Name: sonstigeEigenschaft

Definition: From the AAA 7.1 specification:

"Sonstige Eigenschaft" enthält Informationen zum Netzpunkt. Sonstige Eigenschaften werden im Rahmen der Migration aus bestehenden Verfahrenslösungen übernommen.

Voidable: false

Multiplicity: 0..*

Value type: CharacterString

Attribute:

Name: vermarkung_Marke

Definition: From the AAA 7.1 specification:

'Vermarkung (Marke)' ist die Marke zur dauerhaften Kennzeichnung von Vermessungspunkten im Boden und an baulichen Anlagen.

Voidable: false

Multiplicity: 1

Value type: AX_Marke (enumeration)

Values

1000	**Marke, allgemein**
1100	**Stein**
1111	**Lochstein**
1112	**Vermessungspunktstein**
1120	**Unbehauener Feldstein**
1130	**Gemeinde- und Waldgrenzstein**
1131	**Gemeindegrenzstein**
1132	**Waldgrenzstein, Forstgrenzstein**
1140	**Kunstoffmarke**
1160	**Landesgrenzstein**
1190	**Stein mit Besonderheiten in Form oder Material**
1200	**Rohr**
1201	**Rohr mit Schutzkappe**
1202	**Rohr mit Kopf**
1203	**Rohr mit Bolzen, oberirdisch**
1210	**Eisenrohr**
1211	**Eisenrohr (mit Schutzkappe)**
1212	**Eisenrohr (ohne Schutzkappe)**
1220	**Kunststoffrohr**
1221	**Kunststoffrohr (mit Schutzkappe)**
1222	**Kunststoffrohr (ohne Schutzkappe)**
1230	**Drainrohr**
1240	**Rohr mit Schutzkasten**
1250	**Zementrohr**

1260	**Glasrohr**
1290	**Tonrohr**
1300	**Bolzen/Nagel**
1310	**Bolzen**
1311	**Adapterbolzen**
1320	**Nagel**
1400	**Meisselzeichen (z.B. Kreuz, Kerbe, Anker)**
1410	**Bohrloch**
1500	**Pfahl**
1600	**Sonstige Marke**
1610	**Marke in Schutzberhälter**
1620	**Flasche**
1630	**Platte**
1631	**Klinkerplatte**
1632	**Granitplatte**
1635	**Platte mit Loch**
1640	**Hohlziegel**
1650	**Klebemarke**
1655	**Schlagmarke**
1660	**Kanaldeckel (Kreuz des Gütesiegels auf Rand)**
1670	**Marke besonderer Ausführung**
1700	**Punkt dauerhaft und gut erkennbar festgelegt**
1710	**Punkt der baulichen Anlage**
1711	**Sockel (roh)**
1712	**Sockel (verputzt)**
1713	**Mauerecke (roh)**
1714	**Mauerecke (verputzt)**
1720	**Grenzsaeule**
1800	**Pfeiler**
1820	**Kegel**

	2100	**Festlegung 1. Ordnung, Kopf 30x30 cm, Bezugspunkt Platte**
	2101	**Festlegung 1. Ordnung, Bezugspunkt Kopf 30x30 cm**
	2102	**Festlegung STN 1. Ordnung, Pfeilerkopf 30x30 cm, Bezugspunkt Platte 60x60 cm, Steinwürfel, Tonkegel**
	2110	**Festlegung 2. bis 5. Ordnung, Kopf 16x16 oder 12x12 cm, Bezugspuknt Platte 30x30 cm**
	2111	**Festlegung 2. bis 5. Ordnung, Bezugspunkt Kopf 16x16 oder 12x12 cm, Platte 30x30 cm**
	2120	**Festlegung 2. bis 4. Ordnung, Kopf 20x20 cm, Bezugspunkt Platte**
	2121	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 20x20 cm**
	2130	**Festlegung 2. bis 4. Ordnung, Kopf 25x25 cm, Bezugspunkt Platte**
	2131	**Festlegung 2. bis 4. Ordnung, Bezugspunkt Kopf 25x25 cm**
	2132	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Platte 30x30-40x40 cm**
	2133	**Festlegung STN 3. und 5. Ordnung, Pfeilerkopf 16x16 cm, Bezugspunkt Bolzen oder Rotgusskappe im Fels**
	2134	**Festlegung TP-Feld, Pfeilerkopf 25x25 cm, Bezugspunkt Platte 35x35 cm**
	2135	**Festlegung RBP-Feld, Pfeilerkopf 16x16 cm mit Gravur "TP" und "Dreieck", Bezugspunkt Platte 30x30-35x35 cm**
	2140	**Plattformbolzen mit Aufschrift TP**
	2150	**Turmbolzen mit Aufschrift TP**
	2160	**Leuchtschraube oder -bolzen**
	2161	**Schraube (vertikal)**
	2162	**Messingmarke oder Messingbolzen**

2163	**Keramikbolzen, oberirdisch**
2164	**Bolzen im Fels, unterirdisch**
2165	**Rotgusskappe im Fels, unterirdisch**
2166	**Messingbolzen (gewölbt), Aufschrift TP und Dreieck**
2167	**3D-Messingbolzen (Durchmesser 5 cm) mit Inschrift RFP HESSEN**
2170	**Turmbolzen, Festlegungsbolzen oder sonstiger Bolzen, keine weiteren Angaben bekannt oder gespeichert**
2180	**Festlegung 2. Ordnung, Kopf 16x16 cm oder 12x12 cm, Bezugspunkt Platte 60x60 cm**
2181	**Festlegung 2. Ordnung, Bezugspunkt Kopf 16x16 cm, oder 12x12 cm, Platte 60x60 cm**
2190	**Festlegung 2. bis 3. Ordnung, Kopf 16x16 cm oder 12x12 cm, Platte 40x40 cm**
2191	**Festlegung 2. bis 3. Ordnung, Bezugspunkt Kopf 16x16 cm, Platte 40x40 cm**
2192	**Bergpfeiler ca. 60 cm lang, ohne Platte, Kopf 12x12 cm**
2200	**Pfeiler mit Aufschrift AP**
2201	**Pfeiler mit Aufschrift AP mit Platte, Bezugspunkt Kopf**
2210	**Plattformbolzen mit der Aufschrift AP**
2220	**Turmbolzen mit der Aufschrift AP**
2230	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 1**
2240	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 2**
2250	**Festlegung der Wasserstrassenverwaltung, Stein mit Rohr und Stehbolzen, Typ 3**

	2260	**Festlegung Sachsen-Anhalt SANREF (unterirdischer Granitpfeiler mit Kopfbolzen)**
	2261	**3D-Bolzen einzementiert in Kopfflaeche einer Festlegung STN 1. Ordnung**
	2262	**3D-Bolzen in Kopfflaeche einer Festlegung TP-Feld**
	2263	**3D-Bolzen in Kopffläche einer AF Sachsen 3. Ordnung**
	2264	**3D-Bolzen in Kopfflaeche einer AF Sachsen 5. Ordnung**
	2265	**3D-Bolzen in Granitplatte unter Schutzkasten oder Kunststoff-Schutzrohr**
	2266	**3D-Bolzen in Kopffläche, einbetonierter Granitpfeiler 40x40x90 cm**
	2267	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 25x25x100 cm**
	2268	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 30x30x100 cm**
	2269	**3D-Bolzen in Kopfflaeche, einbetonierter Granitpfeiler 50x50x100 cm**
	2300	**Alte Festlegung der Kgl. Generalkomission und von Kurhessen, Rillenstein**
	2310	**Alte Festlegung von Nassau, exz. ehemals 2.O.**
	2320	**Alte Festlegung von Nassau, exz. ehemals 3.O.**
	2330	**Alte Festlegung von Hessen-Darstadt, 3.O.**
	2340	**Alte Festlegung von Hessen-Darmstadt, 4.O.**
	2350	**Alte Festlegung von Hessen-Darmstadt, 3.O. exz.**

	2360	**Alte Festlegung von Hessen-Darmstadt, 4.O. exz.**
	2370	**Alte Festlegung von Westfalen, 2. O. exz.**
	2400	**Alte Festlegung von Württemberg, exz.**
	2410	**Alte Festlegung von Württemberg**
	2420	**Alte Festlegung von Mecklenburg, ehemals 1. bis 2. Ordnung**
	2430	**Alte Festlegung von Mecklenburg 2. bis 3. Ordnung**
	2440	**Alte Festlegung von Mecklenburg, ehemals 4. Ordnung**
	2450	**Alte Festlegung Sachsen, 1. und 2. Ordnung (Nagelsche Saeule)**
	2460	**Alte Festlegung Sachsen, 3. Ordnung, Pfeilerkopf 35x35 cm mit zentrischer Messingmarke**
	2470	**Alte Festlegung Sachsen, 5. Ordnung, Pfeilerkopf 25x25 cm mit zentrischer Mesingmarke**
	2500	**Alte Festlegung von Baden, exz., Typ 1**
	2510	**Alte Festlegung von Baden, exz., Typ 2**
	2540	**Alte Festlegung von Baden**
	2550	**Rohr mit Schutzkasten, Grundständer**
	2551	**Grundständer mit Messingmarke unter Schutzkasten**
	2552	**Marke (allgemein) unter Schutzkasten**
	2553	**Platte unter Schutzkasten**
	2560	**Rohr mit Verschlusskappe und aufgesetztem Lochstein**
	2600	**Alte Festlegung in Bayern, exz.**
	2601	**Gebohrter Granitstein (Bezusgpunkt) ueber Tonrohr**
	2602	**Gebohrter Granitstein ueber Tonrohr (Bezugspunkt)**

2603	**Gebohrter Granitstein (Bezugspunkt) ueber Platte**
2604	**Gebohrter Granitstein ueber Platte (Bezugspunkt)**
2605	**Gebohrter Granitstein (Bezugspunkt) ueber Eisenrohr**
2606	**Gebohrter Granitstein ueber Eisenrohr (Bezugspunkt)**
2607	**Betonstein (Bezugspunkt) ueber Platte**
2608	**Betonstein ueber Platte (Bezugspunkt)**
2609	**Betonstein mit durchgehendem Bohrloch (Einschaltpunkt-Stein in Bayern)**
2610	**Alte Festlegung in Bayern**
2611	**KT-Stein der oesterreichischen Katastraltriangulation**
2612	**Stein (Bezugspunkt) ueber Platte**
2613	**Stein ueber Platte (Bezugspunkt)**
2614	**Stein (Bezugspunkt) ueber Tonrohr**
2615	**Stein ueber Tonrohr (Bezugspunkt)**
2616	**Stein mit Eisenrohr**
2620	**Alte Festlegung in Sachsen-Coburg, 3.O.**
2630	**Alte Festlegung in Sachsen-Coburg, 4.O.**
2640	**Alte Festlegung in Bayern un der Pfalz, HDNP-Stein**
2700	**Festlegung MP-Pfeiler**
2710	**Festlegung Orientierungspunkt, Bezugspunkt Platte**
2750	**Steinpfeiler**
2760	**Betonpfeiler**
2770	**Kreuz (gemeisselt)**
2800	**Knopf**
2810	**Mitte**

2820	**Spitze**
2830	**Kreuz (Mitte)**
2840	**Helmstange**
2850	**Fahnenstange**
2860	**Wetterstange**
2870	**Blitzableiter**
2880	**Antenne**
2890	**Rohrstange**
2900	**Platte, unterirdisch**
2901	**Platte einbetoniert mit zentrischem Messinbolzen**
2902	**Platte einbetoniert mit zentrischem Bohrloch**
2903	**Platte eibetoniert mit zentrischer Keramikmarke**
2904	**Platte eibetoniert mit zentrischer Messingmarke**
2910	**Steinwuerfel, unterirdisch**
2920	**Steinplatte, unterirdisch**
2930	**Platte, unterirdisch, 60x60 cm**
2940	**Platte, unterirdisch, 30x30 cm**
2950	**Platte, unterirdisch, mit Stehniet**
2951	**Platte, unterirdisch, mit Kopfbolzen**
2960	**Platte, unterirdisch, mit Schutzrohr**
2970	**Pfeiler, 30x40x90 cm, mit Stehniet**
2980	**Platte, mit Bolzen, unterirdisch, im Schacht**
3000	**Unterirdische Festlegung**
3011	**Unterirdische Festlegung mit Achatkugel**
3012	**Unterirdische Festlegung mit Halbkugel**
3013	**Unterirdische Festlegung mit Diabaeinsatz**
3014	**Unterirdische Festlegung im Schacht**

3015	**Kleine unterirdische Festlegung**
3020	**Unterirdischer Rammpfahl**
3030	**Unterirdischer Pfeilerbolzen**
3040	**Unterirdischer Bolzen**
3050	**Hamburger Flachpunkt**
3060	**Unterirdische Saeule**
3070	**Unterirdischer Rammstab**
3100	**Rohrfestpunkt**
3110	**Rohrfestpunkt, Hamburger Bauart**
3120	**Rohrfestpunkt, Oldenburger Bauart**
3130	**Rohrfestpunkt, Eider Bauart**
3140	**Rohrfestpunkt Nordrhein-Westfalen**
3150	**Rohrfestpunkt Nebenpunkt, flach gegründet**
3160	**Rohrfestpunkt, Celler Bauart**
3200	**Mauerbolzen**
3210	**Mauerbolzen, horizontale eingebracht**
3220	**Mauerbolzen, vertikal eingebracht**
3230	**Hoehenmarke**
3231	**Hoehenmarke (Koeniglich Preussische Landesaufnahme)**
3232	**Hoehenmarke (Koeniglich Saechsische Landesaufnahme)**
3233	**Hoehenmarke (Reichsamt fuer Landesaufnahme)**
3240	**Kugelbolzen**
3250	**Tonnenbolzen**
3260	**Landeshoehenbolzen**
3270	**Stehbolzen bzw. Bolzen vertikal**
3280	**Stehniete**
3290	**Sonstiger horizontaler Bolzen**
3300	**Pfeilerbolzen**
3301	**Pfeilerbolzen, Bezugspunkt Pfeileroberflaeche**

	3310	**Pfeilerbolzen, Naturstein, Bolzen horizontal**
	3311	**Pfeilerbolzen, Naturstein, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**
	3320	**Pfeilerbolzen, Naturstein, Bolzen vertikal**
	3330	**Pfeilerbolzen, Beton, Bolzen, horizontal**
	3331	**Pfeilerbolzen, Beton, Bolzen horizontal, Bezugspunkt Pfeileroberflaeche**
	3350	**Pfeilerniete, Naturstein, Niete vertikal**
	3400	**Rammpfahl**
	3410	**Rammpfahl, Bolzen horizontal**
	3420	**Rammpfahl, Bolzen vertikal**
	3810	**Schraubpfahl**
	3820	**Hektometerstein**
	3830	**Markstein**
	3840	**Schraubbolzen**
	3845	**Schraubeisen**
	3850	**Lochmarke/-bolzen (ohne Hoehentafel)**
	3860	**Lochmarke/-bolzen mit Hoehentafel**
	3870	**Festpunktstein**
	3880	**Eichpfahl**
	4100	**Gravimeterplatte 80x80 oder 60x60 cm**
	4110	**Gravimeternagel**
	4120	**Gravimeterpfeiler 20x20x100 cm**
	4130	**Gravimeterpfeiler 16x16x60 cm**
	4140	**Messingscheibe mit zentrischer Woelbung (Durchm. 8 cm)**
	4150	**Stehniete, Messing (Durchmesser 3cm, Aufschrift SFP)**
	4160	**Messplakette, Aufschrift**

	5100	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Betonpfeiler mt Fudament im festen Erdboden)**
	5150	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Betonpfeiler mt Fudament im festen Erdboden)**
	5200	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Gemauerter Pfeiler auf einem Bauwerk)**
	5250	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Gemauerter Pfeiler auf einem Bauwerk)**
	5300	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Stahlpfeiler auf einem Bauwerk)**
	5350	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Stahlpfeiler auf einem Bauwerk)**
	5400	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Seitlich befestigtes Stahlrohr am Bauwerk)**
	5450	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Seitlich befestigtes Stahlrohr am Bauwerk)**
	5500	**Gewindebolzen (hoechste Stelle, Mitte) auf Metallplatte (Antennentraeger)**
	5550	**Oberflaeche der Metallplatte (hoechste Stelle, Mitte) (Antennentraeger)**
	5900	**GNSS-Antennenhalterung, Oberflaeche Platte (Loch) (SL)**
	9000	**Marke unter 'Bemerkung' naeher definiert**
	9500	**Ohne Marke**
	9600	**Abmarkung zeitweilig ausgesetzt**
	9998	**Nach Quellenlage nicht zu spezifizieren**
	9999	**Sonstiges**

Attribute:

Name: zustaendigeStelle

Definition:	From the AAA 7.1 specification: 'Zuständige Stelle' enthält den Namen der Stelle, die eine Zuständigkeit besitzt.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Dienststelle_Schluessel (data type)

1.9.25 AX_Person

AX_Person

Definition:	From the AAA 7.1 specification: 'Person' ist eine natürliche oder juristische Person und kann z.B. in den Rollen Eigentümer, Erwerber, Verwalter oder Vertreter in Katasterangelegenheiten geführt werden.
Subtype of:	LA_Party
Type:	Feature type

Attribute:

Name:	akademischerGrad
Definition:	From the AAA 7.1 specification: 'Akademischer Grad' ist der akademische Grad der Person (z.B. Dipl.-Ing., Dr., Prof. Dr.).
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	anrede
Definition:	From the AAA 7.1 specification: 'Anrede' ist die Anrede der Person. Diese Attributart ist optional, da Körperschaften und juristischen Person auch ohne Anrede angeschrieben werden können.
Voidable:	false
Multiplicity:	0..1
Value type:	AX_Anrede_Person (enumeration)

Values

1000	**Frau**
2000	**Herr**
3000	**Firma**

Attribute:	
Name:	geburtsdatum
Definition:	From the AAA 7.1 specification: 'Geburtsdatum' ist das Geburtsdatum der Person.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	geburtsname
Definition:	From the AAA 7.1 specification: 'Geburtsname' ist der Geburtsname der Person.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	nachnameOderFirma
Definition:	From the AAA 7.1 specification: 'Nachname oder Firma' ist - bei einer natürliche Person der Nachname (Familienname), - bei einer juristischen Person, Handels- oder Partnerschaftsgesellschaft der Name oder die Firma.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	namensbestandteil
Definition:	From the AAA 7.1 specification: 'Namensbestandteil' enthält z.B. Titel wie 'Baron'.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	rufname

Definition:	From the AAA 7.1 specification: 'Rufname' ist der Rufname/ sind die Rufnamen einer natürlichen Person.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	sonstigeEigenschaften
Definition:	From the AAA 7.1 specification: 'Sonstige Eigenschaften' sind weitere die Person deutlich kennzeichnende Merkmale (Par. 15 Grundbuchverfügung). Diese Attributart kommt nur bei Personen vor, die die Rolle 'Eigentümer' besitzen.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	sterbedatum
Definition:	From the AAA 7.1 specification: 'Sterbedatum' ist das Sterbedatum der Person.
Voidable:	false
Multiplicity:	0..1
Value type:	Date
Attribute:	
Name:	vorname
Definition:	From the AAA 7.1 specification: 'Vorname' ist der Vorname/ sind die Vornamen einer natürlichen Person.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Association role	
Name:	wirdVertretenVon
Voidable:	false

Multiplicity:	0..*
Value type:	AX_Vertretung (feature type)
Attribute:	
Name:	wohnortOderSitz
Definition:	From the AAA 7.1 specification: 'Wohnort oder Sitz' ist der Wohnort oder der Sitz einer natürlichen oder juristischen Person (Par. 15 Grundbuchverfügung). Diese Attributart kommt nur bei Personen vor, die die Rolle 'Eigentümer' besitzen.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.9.26 AX_PersonenGruppe

AX_PersonenGruppe	
Definition:	From the AAA 7.1 specification: ''Personengruppe' ist die Zusammenfassung von Personen unter einem Ordnungsbegriff. Die Information wird nach Einführung des Datenbankgrundbuches (DaBaG) von der Grundbuchverwaltung nicht mehr übermittelt.
Subtype of:	LA_GroupParty
Type:	Feature type
Attribute:	
Name:	nameDerPersonengruppe
Definition:	From the AAA 7.1 specification: 'Name der Personengruppe' ist ein Ordnungsbegriff, unter dem Personen zusammengefasst sind.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

1.9.27 AX_Pfeilerhoehe_Lagefestpunkt

AX_Pfeilerhoehe_Lagefestpunkt	
Definition:	From the AAA 7.1 specification:

	Die 'Pfeilerhöhe' gibt bei Vermarkungen, die aus Pfeiler und Platte bestehen, die Höhendifferenz Pfeileroberfläche minus Plattenoberfläche sowie das Messdatum an.
Type:	Data type
Attribute:	
Name:	abstand
Definition:	From the AAA 7.1 specification: Abstand Pfeileroberfläche minus Plattenoberfläche in Millimeter.
Voidable:	false
Multiplicity:	1
Value type:	Length
Attribute:	
Name:	messung
Definition:	From the AAA 7.1 specification: Tag, Monat und Jahr der Messung.
Voidable:	false
Multiplicity:	1
Value type:	Date

1.9.28 AX_Punktstabilitaet_Hoehenfestpunkt

AX_Punktstabilitaet_Hoehenfestpunkt	
Definition:	From the AAA 7.1 specification: 'Punktstabilität' gibt die vermutete bzw. nachgewiesene Höhenstabilität der Punktvermarkung in acht Einzelinformationen an.
Type:	Data type
Attribute:	
Name:	geologischeStabilitaet
Definition:	From the AAA 7.1 specification: 'Geologische Stabilität' gibt die Einflussgröße auf die Stabilität des HFP an.
Voidable:	false
Multiplicity:	0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_GeologischeStabilitaet (enumeration)

Values

1000	**Sehr gut**
2000	**Gut**
3000	**Befriedigend**
4000	**Ausreichend**
5000	**Mangelhaft**
9998	**Nicht untersucht**

Attribute:

Name: grundwasserschwankung

Definition: From the AAA 7.1 specification:

'Grundwasserschwankung' gibt die Einflussgröße auf die Stabilität des HFP an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_Grundwasserschwankung (enumeration)

Values

1000	**Sehr gering (kleiner gleich 0,5 m)**
2000	**Gering (groesser 0,5 m und kleiner gleich 2 m)**
3000	**Maessig (groesser 2 m und kleiner gleich 5 m)**
4000	**Stark (groesser 5 m und kleiner gleich 10 m)**
5000	**Sehr stark (groesser 10 m)**
9998	**Nicht untersucht**

Attribute:

Name: grundwasserstand

Definition: From the AAA 7.1 specification:

'Grundwasserstand' gibt die Einflussgröße auf die Stabilität des HFP an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_Grundwasserstand (enumeration)

Values	1000	**Sehr tief (groesser 10 m)**
	2000	**Tief (groesser 5 m und kleiner gleich 10 m)**
	3000	**Normal (groesser 2 m und kleiner gleich 5 m)**
	4000	**Hoch (groesser 0,5 m und kleiner gleich 2 m)**
	5000	**Sehr hoch (kleiner gleich 0,5 m)**
	9000	**Abgesenkt**
	9998	**Nicht untersucht**

Attribute:

Name: gueteDesBaugrundes

Definition: From the AAA 7.1 specification:

'Güte des Baugrundes' gibt die Einflussgröße auf die Stabilität des HFP an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_GueteDesBaugrundes (enumeration)

Values

1000	**Sehr gut**
2000	**Gut**
3000	**Befriedigend**
4000	**Ausreichend**
5000	**Mangelhaft**
9998	**Nicht untersucht**

Attribute:

Name: gueteDesVermarkungstraegers

Definition: From the AAA 7.1 specification:

'Güte des Vermarkungsträgers' gibt die Qualität des Bauwerkes als Punktträger (Unterkellertes Haus, Durchlass, Rammpfahl u.a.) an. Dadurch sind Aussagen zur möglichen Gefährdung der HFP durch Straßenausbau u.ä. und zur Höhenstabilität möglich.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_GueteDesVermarkungstraegers (enumeration)

Values

1000	**Sehr gut**
2000	**Gut**
3000	**Befriedigend**
4000	**Ausreichend**
5000	**Mangelhaft**
9998	**Nicht bekannt**

Attribute:

Name: hoehenstabilitaetAusWiederholungsmessungen

Definition: From the AAA 7.1 specification:

'Höhenstabilität aus Wiederholungsmessungen' gibt die Höhenstabilität eines Punktes aus Wiederholungsmessungen wieder.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_HoehenstabilitaetAusWiederholungsmessungen (enumeration)

Values

1000	**Sehr gut**
2000	**Gut**
3000	**Befriedigend**
4000	**Bedenklich**
5000	**Mangelhaft**
9998	**Nicht bekannt**

Attribute:

Name: topographieUndUmwelteinfluesse

Definition: From the AAA 7.1 specification:

'Topographie und Umwelteinflüsse' gibt die entsprechenden Einflussgrößen auf die Stabilität des HFP an.

Voidable: false

Multiplicity: 0..1

Value type: AX_PunktstabilitaetHoehenfestpunkt_TopographieUndUmwelt (enumeration)

Values

1000	**Keine**

2000	**Geringe**
3000	**Maessige**
4000	**Starke**
5000	**Sehr starke**
9998	**Nicht untersucht**

Attribute:

Name: vermutetHoehenstabilitaet

Definition: From the AAA 7.1 specification:

'Vermutete Höhenstabilität' gibt die vermutete Höhenstabilität der Punkt-Vermarkung an.

Voidable: false

Multiplicity: 0..1

Value type: AX_Punktstabilitaet_Hoehenfestpunkt_VermuteteHoehenstabilitaet (enumeration)

Values

1000	**Sehr gut**
2000	**Gut**
3000	**Befriedigend**
4000	**Ausreichend**
5000	**Mangelhaft**
5100	**Mangelhaft (Bergsenkungsgebiet)**
5200	**Mangelhaft (in rutschgefaehrdeter Hanglage)**
5300	**Mangelhaft (sehr nahe an Gewaesser)**
5400	**Mangelhaft (instabiler Untergrund)**
9998	**Nicht untersucht**

1.9.29 AX_Sicherungspunkt

AX_Sicherungspunkt

Definition: From the AAA 7.1 specification:

'Sicherungspunkt' ist ein Punkt des Netzpunktfeldes, der vermarkt ist und der Sicherung eines Aufnahmepunktes oder Sonstigen Vermessungspunktes dient.

Subtype of: AX_Netzpunkt

Type: Feature type

1.9.30 AX_SonstigeEigenschaften_Flurstueck

AX_SonstigeEigenschaften_Flurstueck

Definition:	From the AAA 7.1 specification: Der Datentyp setzt sich zusammen aus: 1. Kennung, Schlüssel gemäß Festlegung im ALB 2. Fläche des Abschnitts in Quadratmeter 3. Angaben zum Abschnitt/Flurstück (unstrukturiert) 4. Angaben zum Abschnitt - Stelle 5. Angaben zum Abschnitt - Nummer, Aktenzeichen 6. Angaben zum Abschnitt - Bemerkung, Die Angaben zum Abschnitt/Flurstück sind unstrukturiert (3. Stelle) oder strukturiert (4. - 6. Stelle).
Type:	Data type

Attribute:

Name:	angabenZumAbschnittBemerkung
Definition:	From the AAA 7.1 specification: Angaben zum Abschnitt - Bemerkung.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	angabenZumAbschnittFlurstueck
Definition:	From the AAA 7.1 specification: Angaben zum Abschnitt/Flurstück (unstrukturiert).
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	angabenZumAbschnittNummerAktenzeichen
Definition:	From the AAA 7.1 specification: Angaben zum Abschnitt - Nummer, Aktenzeichen.
Voidable:	false
Multiplicity:	0..1

Value type:	CharacterString
Attribute:	
Name:	angabenZumAbschnittStelle
Definition:	From the AAA 7.1 specification: Angaben zum Abschnitt - Stelle.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	flaecheDesAbschnitts
Definition:	From the AAA 7.1 specification: Fläche des Abschnitts bzw. Flurstück in Quadratmeter.
Voidable:	false
Multiplicity:	0..1
Value type:	Area
Attribute:	
Name:	kennungSchluessel
Definition:	From the AAA 7.1 specification: 'Kennung, Schlüssel' gemäß der Festlegung im ALB.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.9.31 AX_SonstigerVermessungspunkt

AX_SonstigerVermessungspunkt	
Definition:	From the AAA 7.1 specification: 'Sonstiger Vermessungspunkt' ist ein Punkt des Aufnahmepunktfeldes, der weder Aufnahmepunkt noch Sicherungspunkt ist (z. B. Polygonpunkt, Liniennetzpunkt).
Subtype of:	AX_Netzpunkt
Type:	Feature type
Attribute:	
Name:	art

Definition:	From the AAA 7.1 specification: 'Art' beschreibt die Art des sonstigen Vermessungspunktes.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Association role	
Name:	mit
Voidable:	false
Multiplicity:	0..*
Value type:	AX_Sicherungspunkt (feature type)

1.9.32 AX_Vertretung

AX_Vertretung	
Definition:	From the AAA 7.1 specification: 'Vertretung' gibt an, welche Person eine andere Person in Katasterangelegenheiten vertritt.
Subtype of:	LA_Party
Type:	Feature type
Attribute:	
Name:	angabenZurVertretung
Definition:	From the AAA 7.1 specification: 'Angaben zur Vertretung' beschreibt den Umfang der Vertretung (z.B. alle Flurstücke einer Gemeinde).
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	artDerVertretung
Definition:	From the AAA 7.1 specification: 'Art der Vertretung' beschreibt die Art der Vertretung (z.B. Bevollmächtigter).
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString

1.10 Package: ConstructionLaw

Definition:

The package ConstructionLaw provides classes abstracting objects in the context of construction law.

Parent package:

Application schema: Siteplan

Diagram(s):

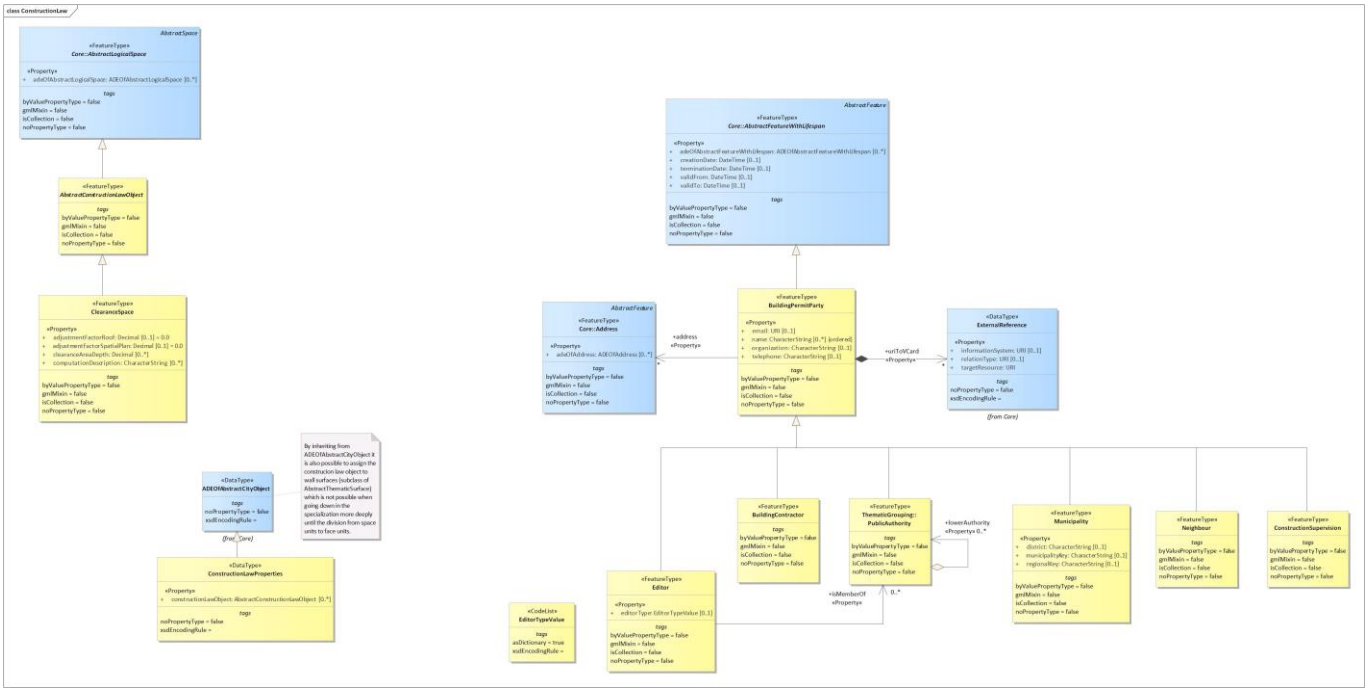


Diagram 33 - ConstructionLaw

1.10.1 AbstractConstructionLawObject

AbstractConstructionLawObject	
Definition:	Abstract class defining a superclass for all construction law objects.
Subtype of:	AbstractLogicalSpace
Supertype of:	ClearanceSpace
Type:	Feature type
Abstract:	true

1.10.2 BuildingContractor

BuildingContractor	
Definition:	The party applying for the building permit.
Subtype of:	BuildingPermitParty

Type:	Feature type

1.10.3 BuildingPermitParty

BuildingPermitParty	
Definition:	An involved party in the building permit process.
Subtype of:	AbstractFeatureWithLifespan
Supertype of:	BuildingContractor
	ConstructionSupervision
	Editor
	Municipality
	Neighbour
	PublicAuthority
Type:	Feature type
Association role	
Name:	address
Voidable:	false
Multiplicity:	0..*
Value type:	Address (feature type)
Attribute:	
Name:	email
Definition:	Email address.
Voidable:	false
Multiplicity:	0..1
Value type:	URI
Attribute:	
Name:	name
Definition:	Names of the building permit party, ordered from surnames to last names.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString
Attribute:	
Name:	organization

Definition:	The name of the organization of the building permit party.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	telephone
Definition:	The telephone number of the building permit party.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Association role	
Name:	uriToVCard
Voidable:	false
Multiplicity:	0..*

1.10.4 ClearanceSpace

ClearanceSpace	
Definition:	Class modelling the clearance space of an object according to the jurisdiction.
Subtype of:	AbstractConstructionLawObject
Type:	Feature type
Attribute:	
Name:	adjustmentFactorRoof
Definition:	Factor which results from adjacent roofs which are considered in the calculation of the clearance space depth.
Voidable:	false
Multiplicity:	0..1
Initial value:	0.0
Value type:	Decimal
Attribute:	
Name:	adjustmentFactorSpatialPlan
Definition:	Factor which results from the devellopment area of the building project which isconsidered in the calculation of the clearance space depth.

Voidable:	false
Multiplicity:	0..1
Initial value:	0.0
Value type:	Decimal
Attribute:	
Name:	clearanceAreaDepth
Definition:	The resulting depth of the clearance space.
Voidable:	false
Multiplicity:	0..*
Value type:	Decimal
Attribute:	
Name:	computationDescription
Definition:	Note describing the derivation of the clearance space depth. Might be provided as a mathematical formula.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString

1.10.5 ConstructionLawProperties

ConstructionLawProperties	
Definition:	Hook to provide construction law objects for each type of city objects.
Subtype of:	ADEOfAbstractCityObject
Type:	Data type
Attribute:	
Name:	constructionLawObject
Definition:	Provided construction law objects which apply to the respective city object.
Voidable:	false
Multiplicity:	0..*
Value type:	AbstractConstructionLawObject (feature type)

1.10.6 ConstructionSupervision

ConstructionSupervision

Definition:	The construction supervision is responsible to ensure that the building project as is realized as it was permitted by the public authorities.
Subtype of:	BuildingPermitParty
Type:	Feature type

1.10.7 Editor

Editor	
Definition:	Architect, Site plan creator, surveyor, worker at a public authority, etc.
Subtype of:	BuildingPermitParty
Type:	Feature type
Attribute:	
Name:	editorType
Definition:	Value of detailed classification of the editor.
Voidable:	false
Multiplicity:	0..1
Value type:	EditorTypeValue (code list)
Association role	
Name:	isMemberOf
Voidable:	false
Multiplicity:	0..*
Value type:	PublicAuthority (feature type)

1.10.8 Municipality

Municipality	
Definition:	The municipality in which the building project should take place.
Subtype of:	BuildingPermitParty
Type:	Feature type
Attribute:	
Name:	district
Definition:	Name of the district.
Voidable:	false

Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	municipalityKey
Definition:	Identifier of the municipality.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	regionalKey
Definition:	Regional identifier.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.10.9 Neighbour

Neighbour	
Definition:	The neighbours/property owners in the close/nearer surrounding of the building project.
Subtype of:	BuildingPermitParty
Type:	Feature type

1.11 Package: RiskAssessment

Definition:

The RiskAssessment package provides classes to represent hazardous zones and physical properties of spatial features (might be used for valuation of real estate properties or insurance premiums).

Parent package:

Application schema: Siteplan

Diagram(s):

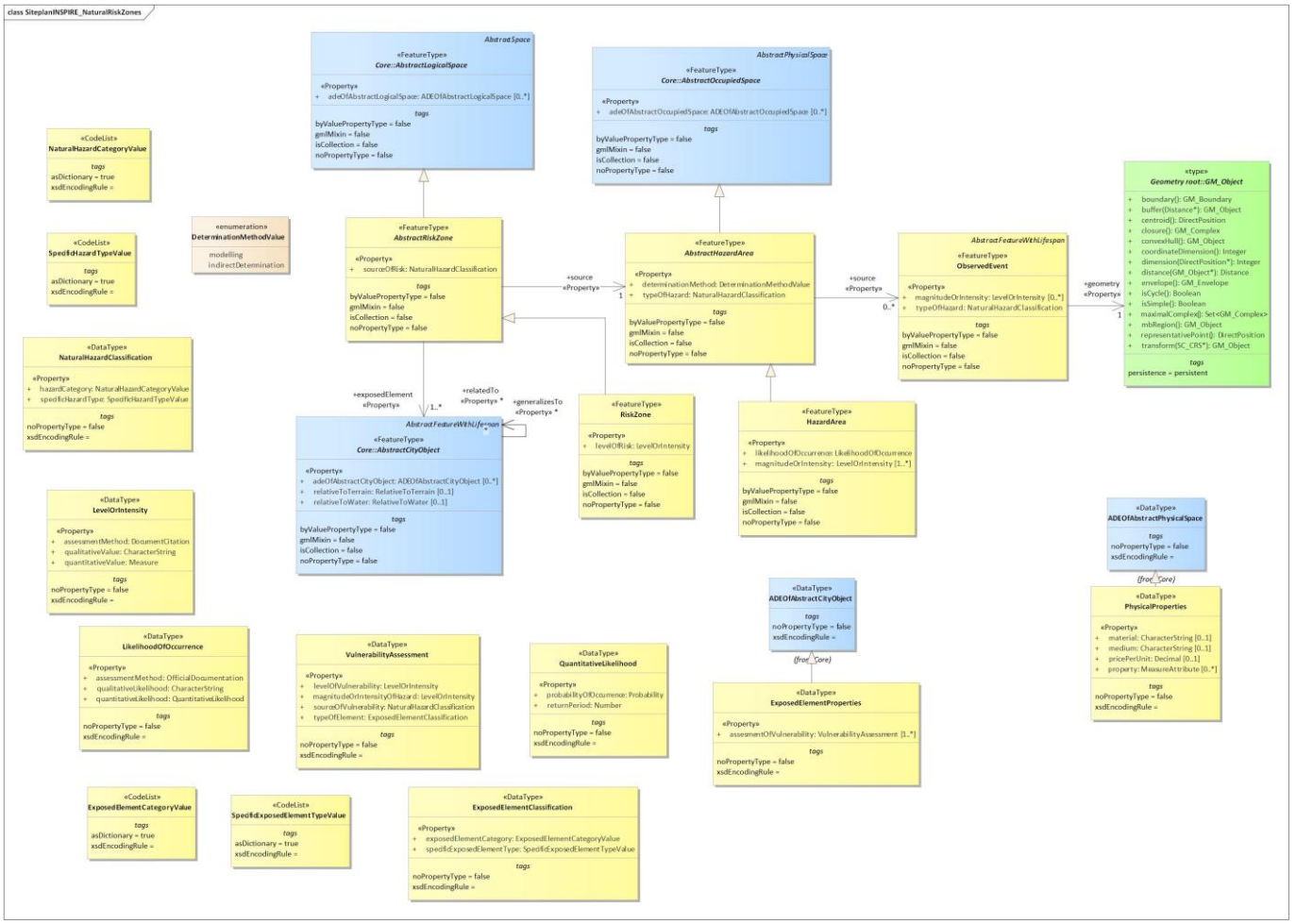


Diagram 34 - SiteplanINSPIRE_NaturalRiskZones

1.11.1 AbstractHazardArea

AbstractHazardArea	
Definition:	From the INSPIRE data specification: An area affected by a natural hazard.
Subtype of:	AbstractOccupiedSpace
Supertype of:	HazardArea
Type:	Feature type
Abstract:	true
Attribute:	
Name:	determinationMethod
Definition:	From the INSPIRE data specification: Specifies if the hazard area result is delineated after modelling or determined after interpretation.
Voidable:	false
Multiplicity:	1

Value type:	DeterminationMethodValue (enumeration)	
Values	modelling	
	indirectDetermination	
Association role		
Name:	source	
Voidable:	false	
Multiplicity:	0..*	
Value type:	ObservedEvent (feature type)	
Attribute:		
Name:	typeOfHazard	
Definition:	From the INSPIRE data specification: A generic classification and a specific classification of the type of natural hazard.	
Voidable:	false	
Multiplicity:	1	
Value type:	NaturalHazardClassification (data type)	

1.11.2 AbstractRiskZone

AbstractRiskZone	
Definition:	From the INSPIRE specification: A risk zone is the spatial extent of a combination of the consequences of an event (hazard) and the associated probability/likelihood of its occurrence.
Subtype of:	AbstractLogicalSpace
Supertype of:	RiskZone
Type:	Feature type
Abstract:	true
Association role	
Name:	exposedElement
Voidable:	false
Multiplicity:	1..*
Value type:	AbstractCityObject (feature type)
Association role	
Name:	source

Voidable:	false
Multiplicity:	1
Value type:	AbstractHazardArea (feature type)
Attribute:	
Name:	sourceOfRisk
Definition:	From the INSPIRE specification: A generic classification and a specific classification of the type of hazard which is the source of risk.
Voidable:	false
Multiplicity:	1
Value type:	NaturalHazardClassification (data type)

1.11.3 CadastralMap

CadastralMap	
Definition:	Grouping of city objects beloning to the cadastral map.
Subtype of:	CityObjectGroup
Type:	Feature type

1.11.4 ExposedElementClassification

ExposedElementClassification	
Definition:	From the INPSIRE specification: This class provides piece of information about the nature of the exposed element which is relevant to risk analysis.
Type:	Data type
Attribute:	
Name:	exposedElementCategory
Definition:	From the INSPIRE specification: A generic classification of the types of elements that are exposed to a risk.
Voidable:	false
Multiplicity:	1
Value type:	ExposedElementCategoryValue (code list)
Attribute:	
Name:	specificExposedElementType

Definition:	From the INSPIRE specifiaction: An additional denomination of exposed element according to a nomenclature that is specific to this dataset.
Voidable:	false
Multiplicity:	1
Value type:	SpecificExposedElementTypeValue (code list)

1.11.5 ExposedElementProperties

ExposedElementProperties	
Definition:	Additional properties for the exposed element.
Subtype of:	ADEOfAbstractCityObject
Type:	Data type
Attribute:	
Name:	assesmentOfVulnerability
Definition:	From the INSPIRE specification: Assessment of the vulnerability of the exposed element.
Voidable:	false
Multiplicity:	1..*
Value type:	VulnerabilityAssessment (data type)

1.11.6 HazardArea

HazardArea	
Definition:	From the INSPIRE data specification: Discrete spatial objects representing a natural hazard.
Subtype of:	AbstractHazardArea
Type:	Feature type
Attribute:	
Name:	likelihoodOfOccurrence
Definition:	From the INSPIRE specification: Likelihood is a general concept relating to the chance of an event occuring.
Voidable:	false
Multiplicity:	1
Value type:	LikelihoodOfOccurrence (data type)

Attribute:	
Name:	magnitudeOrIntensity
Definition:	From the INSPIRE specification: An expression of the magnitude or the intensity of a phenomenon. It may address a value within the Richter scale, or a description of the european macro-seismic scale, or a flood flow, etc...
Voidable:	false
Multiplicity:	1..*
Value type:	LevelOrIntensity (data type)

1.11.7 LevelOrIntensity

LevelOrIntensity	
Definition:	From the INSPIRE data specification: Quantitative or qualitative assessment of either risk, hazard or vulnerability. Common concept for assessing the level of risk, or the level of hazard, or the level of vulnerability.
Type:	Data type
Attribute:	
Name:	assessmentMethod
Definition:	From the INSPIRE specification: A citation to the method used to express the level or intensity.
Voidable:	false
Multiplicity:	1
Value type:	DocumentCitation (data type)
Attribute:	
Name:	qualitativeValue
Definition:	From the INSPIRE specification: A qualitative assessment of the level or intensity.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString

Attribute:	
Name:	quantitativeValue
Definition:	From the INSPIRE specification: A quantitative assessment of the level or intensity.
Voidable:	false
Multiplicity:	1
Value type:	Measure

1.11.8 LikelihoodOfOccurrence

LikelihoodOfOccurrence	
Definition:	From the INSPIRE data specification: Likelihood is a general concept relating to the chance of an event occurring. Likelihood is generally expressed as a probability or a frequency. [EXCIFF].
Type:	Data type
Attribute:	
Name:	assessmentMethod
Definition:	From the INSPIRE specification: A citation to the method used to express the likelihood.
Voidable:	false
Multiplicity:	1
Value type:	OfficialDocumentation (feature type)
Attribute:	
Name:	qualitativeLikelihood
Definition:	From the INSPIRE specification: A qualitative assessment of the likelihood of occurrence of a hazard. Sometimes, this is known as susceptibility.
Voidable:	false
Multiplicity:	1
Value type:	CharacterString
Attribute:	
Name:	quantitativeLikelihood

Definition:	From the INSPIRE specification: A frequency of occurence or return period of a hazard phenomenon. Sometimes, this is known as susceptibility.
Voidable:	false
Multiplicity:	1
Value type:	QuantitativeLikelihood (data type)

1.11.9 NaturalHazardClassification

NaturalHazardClassification	
Definition:	From the INSPIRE data specification: This class provides piece of information about the nature of the natural hazard as well as the type of hazard which is the source of risk
Type:	Data type
Attribute:	
Name:	hazardCategory
Definition:	From the INSPIRE specification: A generic classification of types of natural hazards or risks.
Voidable:	false
Multiplicity:	1
Value type:	NaturalHazardCategoryValue (code list)
Attribute:	
Name:	specificHazardType
Definition:	From the INSPIRE specification: Additional classification of the natural hazard that further specifies the hazard type according to a nomenclature that is specific to this dataset.
Voidable:	false
Multiplicity:	1
Value type:	SpecificHazardTypeValue (code list)

1.11.10 ObservedEvent

ObservedEvent	
Definition:	From the INSPIRE specifiaction:

	Discrete spatial objects representing natural phenomenon relevant to the study of natural hazards which occurred, or is currently occuring, and which has been observed.
Subtype of:	AbstractFeatureWithLifespan
Type:	Feature type
Association role	
Name:	geometry
Voidable:	false
Multiplicity:	1
Value type:	GM_Object (feature type)
Attribute:	
Name:	magnitudeOrIntensity
Definition:	From the INSPIRE specification: An expression of the magnitude or the intensity of a phenomenon. It may address a value within the Richter scale, or a description of the european macro-seismic scale, or a flood flow, etc...
Voidable:	false
Multiplicity:	0..*
Value type:	LevelOrIntensity (data type)
Attribute:	
Name:	typeOfHazard
Definition:	From the INSPIRE data specification: A generic classification and a specific classification of the type of natural hazard.
Voidable:	false
Multiplicity:	1
Value type:	NaturalHazardClassification (data type)

1.11.11 PhysicalProperties

PhysicalProperties	
Definition:	Giving information of the physical properties of the physical space unit which can be used to conduct real estate valuation.
Subtype of:	ADEOfAbstractPhysicalSpace

Type:	Data type
Attribute:	
Name:	material
Definition:	Name of the material of which the physical space consists.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	medium
Definition:	Name of the medium contained by the physical space.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	pricePerUnit
Definition:	Price per unit of the material.
Voidable:	false
Multiplicity:	0..1
Value type:	Decimal
Attribute:	
Name:	property
Definition:	Attribute storing further physical properties.
Voidable:	false
Multiplicity:	0..*
Value type:	MeasureAttribute (data type)

1.11.12 QuantitativeLikelihood

QuantitativeLikelihood	
Definition:	From the INSPIRE data specification: A frequency of occurrence or return period of a hazard phenomenon.
Type:	Data type
Attribute:	

Name:	probabilityOfOccurrence
Definition:	From the INSPIRE specification: The probability of occurrence of a hazard event, expressed as a value between 0 and 1. This is the inverse value of the return period.
Voidable:	false
Multiplicity:	1
Value type:	Probability
Attribute:	
Name:	returnPeriod
Definition:	From the INSPIRE specification: Long-term average interval of time or number of years within which an event will be equalled or exceeded [UNESCO].
Voidable:	false
Multiplicity:	1
Value type:	Number

1.11.13 RiskZone

RiskZone	
Definition:	From the INSPIRE data specification: Discrete spatial objects representing the spatial extent of a combination of the consequences of an event (hazard) and the associated probability/likelihood of its occurrence.
Subtype of:	AbstractRiskZone
Type:	Feature type
Attribute:	
Name:	levelOfRisk
Definition:	From the INSPIRE specification: The level of risk is an assessment of the combination of the consequences of an event (hazard) and the associated probability/likelihood of the occurrence of the event.
Voidable:	false
Multiplicity:	1
Value type:	LevelOrIntensity (data type)

1.11.14 VulnerabilityAssessment

VulnerabilityAssessment	
Definition:	From the INSPIRE data specification: Assessment of the vulnerability. It contains piece of information about the source the vulnerability, about the level of vulnerability and about the magnitude or intensity of the hazard for which vulnerability is assessed.
Type:	Data type
Attribute:	
Name:	levelOfVulnerability
Definition:	From the INSPIRE specification: Level of vulnerability. When assessed quantitatively, it is a percentage.
Voidable:	false
Multiplicity:	1
Value type:	LevelOrIntensity (data type)
Attribute:	
Name:	magnitudeOrIntensityOfHazard
Definition:	From the INSPIRE specification: An expression of the magnitude or the intensity of a phenomenon. It may address a value within the Richter scale, or a description of the european macro-seismic
Voidable:	false
Multiplicity:	1
Value type:	LevelOrIntensity (data type)
Attribute:	
Name:	sourceOfVulnerability
Definition:	From the INSPIRE specification: The type of hazard for which the vulnerability is assessed.
Voidable:	false
Multiplicity:	1
Value type:	NaturalHazardClassification (data type)
Attribute:	

Name:	typeOfElement
Definition:	From the INSPIRE specification: A classification of the exposed element.
Voidable:	false
Multiplicity:	1
Value type:	ExposedElementClassification (data type)

1.12 Package: Building

Definition:

The Building package provides additional property classes to the CityGML-Building package in the context of a site plan.

Parent package:

Application schema: Siteplan

Diagram(s):

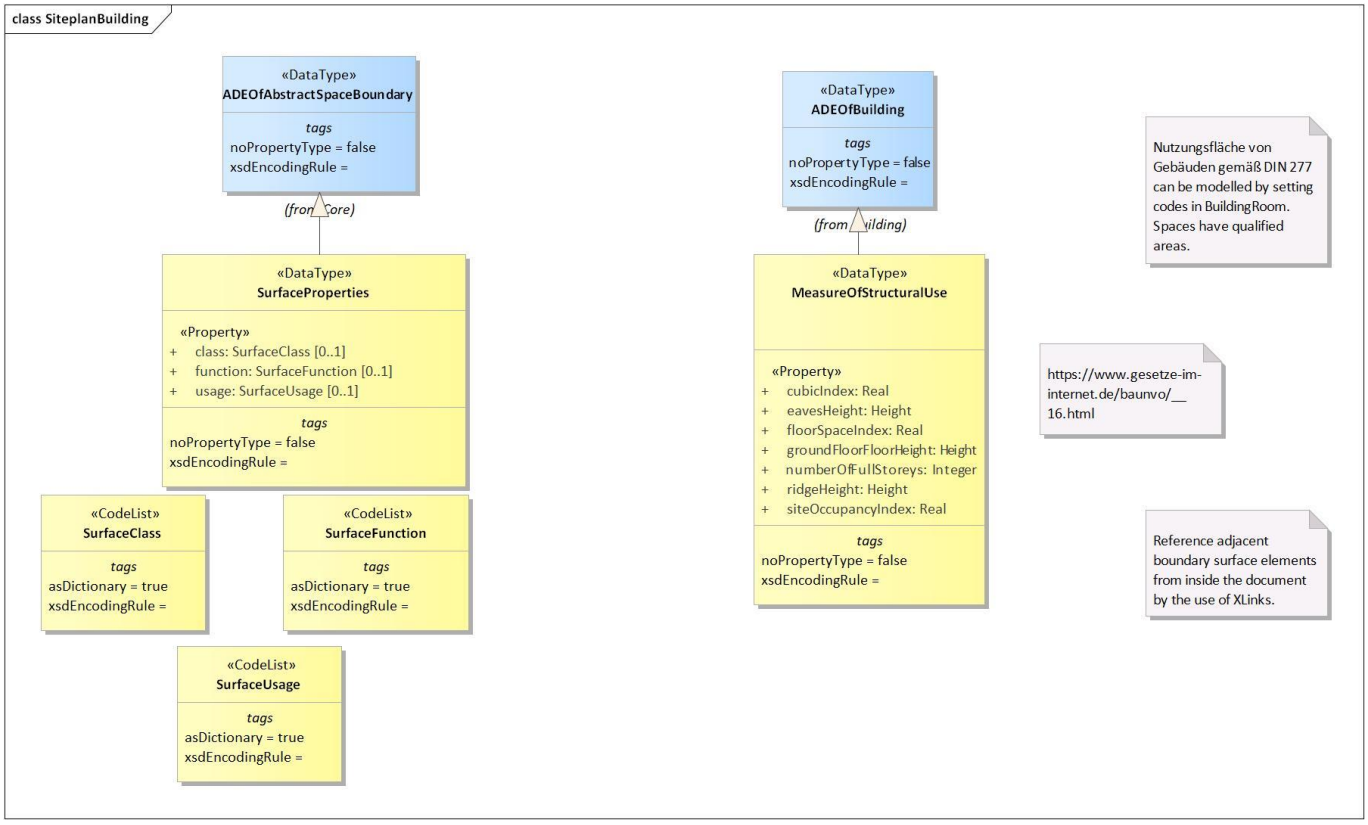


Diagram 35 - SiteplanBuilding

1.12.1 MeasureOfStructuralUse

MeasureOfStructuralUse	
Definition:	Additional properties providing information of the structual use of the building, which can be used to check the legal

	state of the building w.r.t. the regulations defined by the spatial plan.
Subtype of:	ADEOfBuilding
Type:	Data type
Attribute:	
Name:	cubicIndex
Definition:	The building's measure of cubic metres per property area (Baumassenzahl).
Voidable:	false
Multiplicity:	1
Value type:	Real
Attribute:	
Name:	eavesHeight
Definition:	Height of the eave w.r.t. to the reference height (Traufhöhe).
Voidable:	false
Multiplicity:	1
Value type:	Height (data type)
Attribute:	
Name:	floorSpaceIndex
Definition:	The building's area per floor and per property parcel area (Geschossflächenzahl).
Voidable:	false
Multiplicity:	1
Value type:	Real
Attribute:	
Name:	groundFloorFloorHeight
Definition:	Height of the ground floor which can be used as reference height for the eaves height and ridge height.
Voidable:	false
Multiplicity:	1
Value type:	Height (data type)
Attribute:	
Name:	numberOfFullStoreys
Definition:	The building's number of full storeys.

Voidable:	false
Multiplicity:	1
Value type:	Integer
Attribute:	
Name:	ridgeHeight
Definition:	Height of the most upper edge of the roof w.r.t. to the reference height (Firsthöhe).
Voidable:	false
Multiplicity:	1
Value type:	Height (data type)
Attribute:	
Name:	siteOccupancyIndex
Definition:	Computed area of the property parcel which is occupied by the building (Grundflächenzahl).
Voidable:	false
Multiplicity:	1
Value type:	Real

1.12.2 SurfaceProperties

SurfaceProperties	
Definition:	Additional properties of thematic surfaces providing information of the type, function or usage of the surface (e.g. facade surface, side wall, ...).
Subtype of:	ADEOfAbstractSpaceBoundary
Type:	Data type
Attribute:	
Name:	class
Definition:	Indicates the specific type of the thematic surface.
Voidable:	false
Multiplicity:	0..1
Value type:	SurfaceClass (code list)
Attribute:	
Name:	function
Definition:	Specifies the intended purposes of the thematic surface.

Voidable:	false
Multiplicity:	0..1
Value type:	SurfaceFunction (code list)
Attribute:	
Name:	usage
Definition:	Specifies the actual uses of the thematic surface.
Voidable:	false
Multiplicity:	0..1
Value type:	SurfaceUsage (code list)

1.13 Package: RegulationConflict

Definition:

The RegulationConflict package provides the classes to represent a conflict between spatial features and legal regulations.

Parent package:

Application schema: Siteplan

Diagram(s):

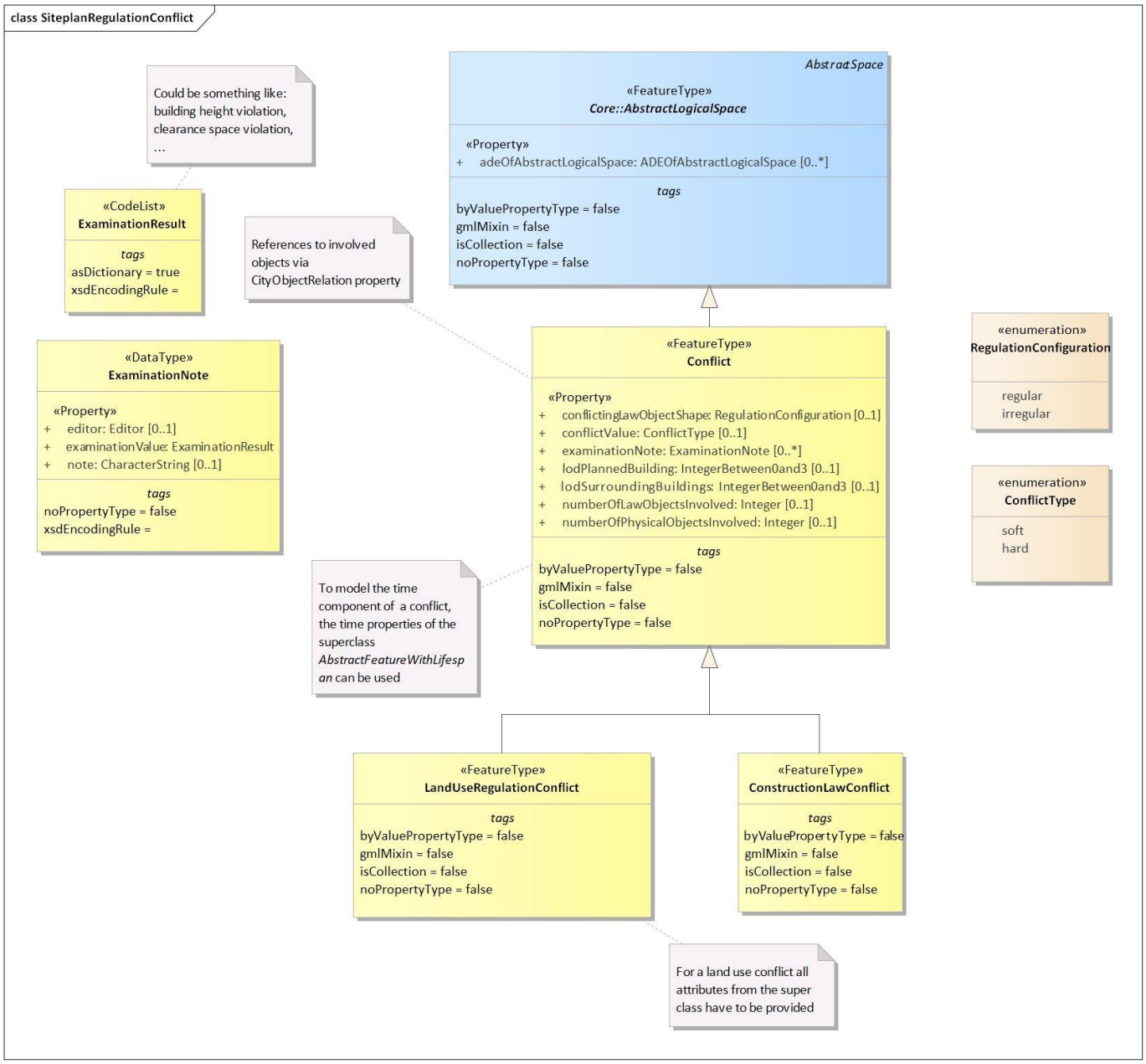


Diagram 36 - SiteplanRegulationConflict

1.13.1 Conflict

Conflict	
Definition:	Describing a spatial conflict between spatial features or between a spatial feature and a spatial/public/private regulation.
Subtype of:	AbstractLogicalSpace
Supertype of:	ConstructionLawConflict LandUseRegulationConflict
Type:	Feature type
Attribute:	

Name: conflictValue

Definition: Class of conflict which is determined by the other of the parameters according to (Emamgholian et al., 2020, MODELLING LAND-USE REGULATOIN CONFLICTS WITH 3D COMPONENTS TO SUPPORT ISSUING A BUILDING PERMIT).

Voidable: false

Multiplicity: 0..1

Value type: ConflictType (enumeration)

Values

soft	
hard	

Attribute:

Name: conflictingLawObjectShape

Definition: Shapes used to model the land-use regulations.

Voidable: false

Multiplicity: 0..1

Value type: RegulationConfiguration (enumeration)

Values

regular	
irregular	

Attribute:

Name: examinationNote

Definition: Additional remarks regarding the examination resulting in the conflict.

Voidable: false

Multiplicity: 0..*

Value type: ExaminationNote (data type)

Attribute:

Name: lodPlannedBuilding

Definition: Level of detail of the planned building.

Voidable: false

Multiplicity: 0..1

Value type: IntegerBetween0and3

Attribute:

Name: lodSurroundingBuildings

Definition:	Level of detail of the surrounding buildings.
Voidable:	false
Multiplicity:	0..1
Value type:	IntegerBetween0and3
Attribute:	
Name:	numberOfLawObjectsInvolved
Definition:	Frequency of conceptual/law features involved in conflict.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer
Attribute:	
Name:	numberOfPhysicalObjectsInvolved
Definition:	Frequency of physical features involved in conflict.
Voidable:	false
Multiplicity:	0..1
Value type:	Integer

1.13.2 ConstructionLawConflict

ConstructionLawConflict	
Definition:	Specification of the conflict superclass determining that the conflict is a construction law conflict.
Subtype of:	Conflict
Type:	Feature type

1.13.3 ExaminationNote

ExaminationNote	
Definition:	Information of the resulting examination of an conflict.
Type:	Data type
Attribute:	
Name:	editor
Definition:	The editor who processed or supervised the examination.
Voidable:	false
Multiplicity:	0..1

Value type:	Editor (feature type)
Attribute:	
Name:	examinationValue
Definition:	Value specifying the examination result.
Voidable:	false
Multiplicity:	1
Value type:	ExaminationResult (code list)
Attribute:	
Name:	note
Definition:	Additional note to add specific remarks on the examination result.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.13.4 LandUseRegulationConflict

LandUseRegulationConflict	
Definition:	Specification of the conflict superclass determining that the conflict is a land use regulation conflict.
Subtype of:	Conflict
Type:	Feature type

1.14 Package: Core

Definition:

The package Core adds properties to the standard classes of the CityGML Core module.

Parent package:

Application schema: Siteplan

Diagram(s):

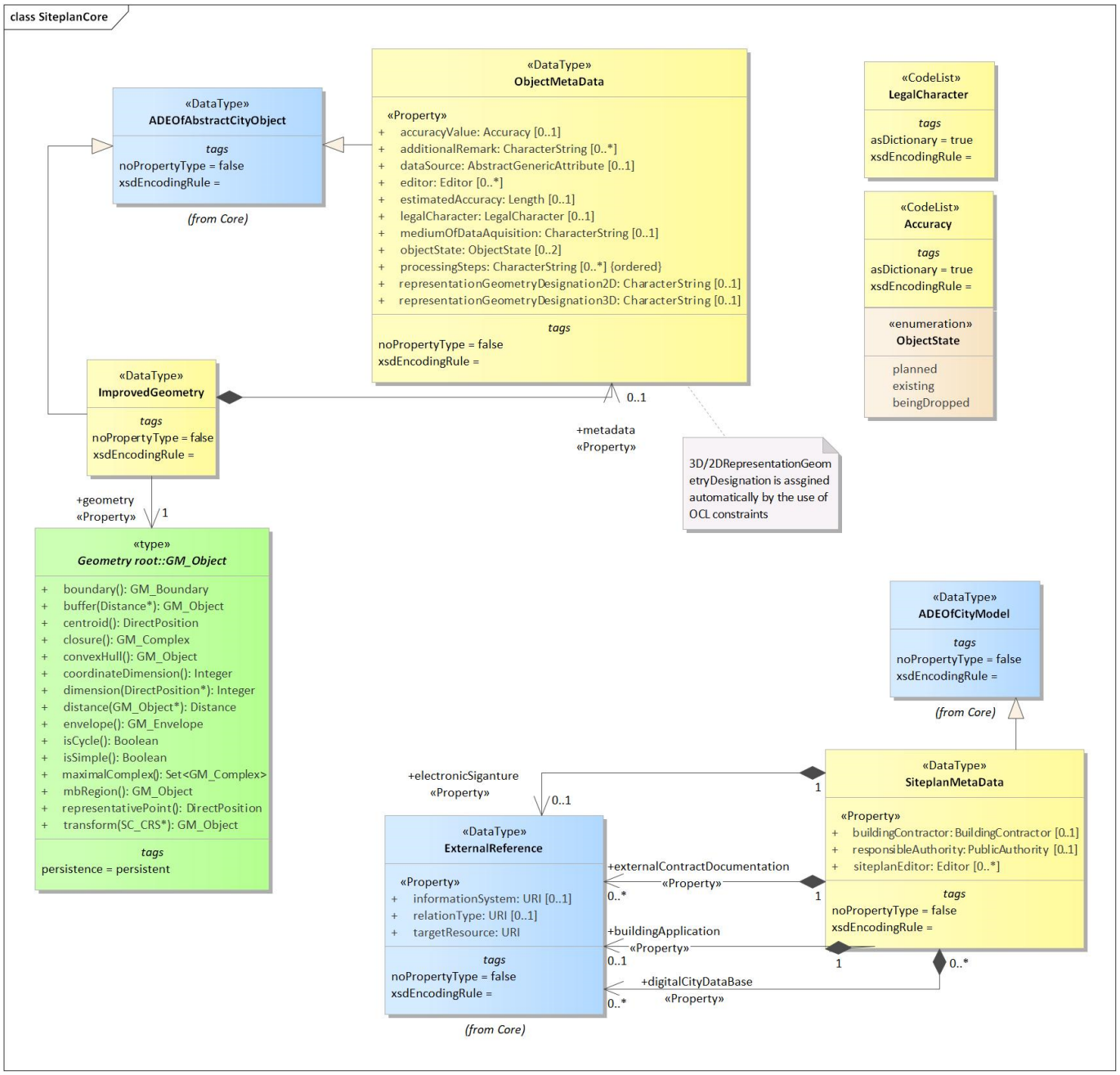


Diagram 37 - SiteplanCore

1.14.1 ImprovedGeometry

ImprovedGeometry	
Definition:	Additional property to provide improved geometry for city objects, which can stem from surveying, etc.
Subtype of:	ADEOfAbstractCityObject
Type:	Data type
Association role	
Name:	geometry
Voidable:	false

Multiplicity:	1
Value type:	GM_Object (feature type)
Association role	
Name:	metadata
Voidable:	false
Multiplicity:	0..1

1.14.2 ObjectMetaData

ObjectMetaData	
Definition:	Additional property to provide meta data for city objects to give legal certainty and transparency of the origin and quality of data the city object.
Subtype of:	ADEOfAbstractCityObject
Type:	Data type
Attribute:	
Name:	accuracyValue
Definition:	A value describing the accuracy of the referred object.
Voidable:	false
Multiplicity:	0..1
Value type:	Accuracy (code list)
Attribute:	
Name:	additionalRemark
Definition:	Additional notes giving information on the respective object.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString
Attribute:	
Name:	dataSource
Definition:	Property holding information on the data source of the object.
Voidable:	false
Multiplicity:	0..1
Value type:	AbstractGenericAttribute (data type)
Attribute:	

Name:	editor
Definition:	The editor is responsible for the processing and contribution of the object.
Voidable:	false
Multiplicity:	0..*
Value type:	Editor (feature type)
Attribute:	
Name:	estimatedAccuracy
Definition:	A quantitative value describing the accuracy (e.g. standard deviation, etc.) of the object's measures.
Voidable:	false
Multiplicity:	0..1
Value type:	Length
Attribute:	
Name:	legalCharacter
Definition:	Describing the legal character of the object (e.g. nachrichtliche Übernahme).
Voidable:	false
Multiplicity:	0..1
Value type:	LegalCharacter (code list)
Attribute:	
Name:	mediumOfDataAquisition
Definition:	Value holding the medium/instrument which was used to create the data for the object.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString
Attribute:	
Name:	objectState
Definition:	Value holding the object's state.
Voidable:	false
Multiplicity:	0..2
Value type:	ObjectState (enumeration)
Values	planned

	existing	
	beingDropped	

Attribute:

Name:	processingSteps
Definition:	Sequence of processing steps which were conducted yielding the end product which result in the object.
Voidable:	false
Multiplicity:	0..*
Value type:	CharacterString

Attribute:

Name:	representationGeometryDesignation2D
Definition:	The attribute name representing the 2D geometry of the object (e.g. lod0MultiSurface), which can be used to derive a 2D representation of the site plan.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

Attribute:

Name:	representationGeometryDesignation3D
Definition:	The attribute name representing the 3D geometry of the object (e.g. lod3Solid), which can be used to derive a 3D representation of the site plan.
Voidable:	false
Multiplicity:	0..1
Value type:	CharacterString

1.14.3 SiteplanMetaData

SiteplanMetaData

Definition:	Meta data concerning the site plan w.r.t. involved parties.
Subtype of:	ADEOfCityModel
Type:	Data type

Association role

Name:	buildingApplication
Voidable:	false

Multiplicity:	0..1
Attribute:	
Name:	buildingContractor
Definition:	The building contractor applying for the building permit.
Voidable:	false
Multiplicity:	0..1
Value type:	BuildingContractor (feature type)
Association role	
Name:	digitalCityDataBase
Voidable:	false
Multiplicity:	0..*
Association role	
Name:	electronicSiganture
Voidable:	false
Multiplicity:	0..1
Association role	
Name:	externalContractDocumentation
Voidable:	false
Multiplicity:	0..*
Attribute:	
Name:	responsibleAuthority
Definition:	The responsible authority processing the building application.
Voidable:	false
Multiplicity:	0..1
Value type:	PublicAuthority (feature type)
Attribute:	
Name:	siteplanEditor
Definition:	The editor responsible for the site plan creation.
Voidable:	false
Multiplicity:	0..*
Value type:	Editor (feature type)

1.15 Package: ThematicGrouping

Definition:

The ThematicGrouping package provides classes for the logical aggregation of site plan objects into specified thematic groupings.

Parent package:

Application schema: Siteplan

Diagram(s):

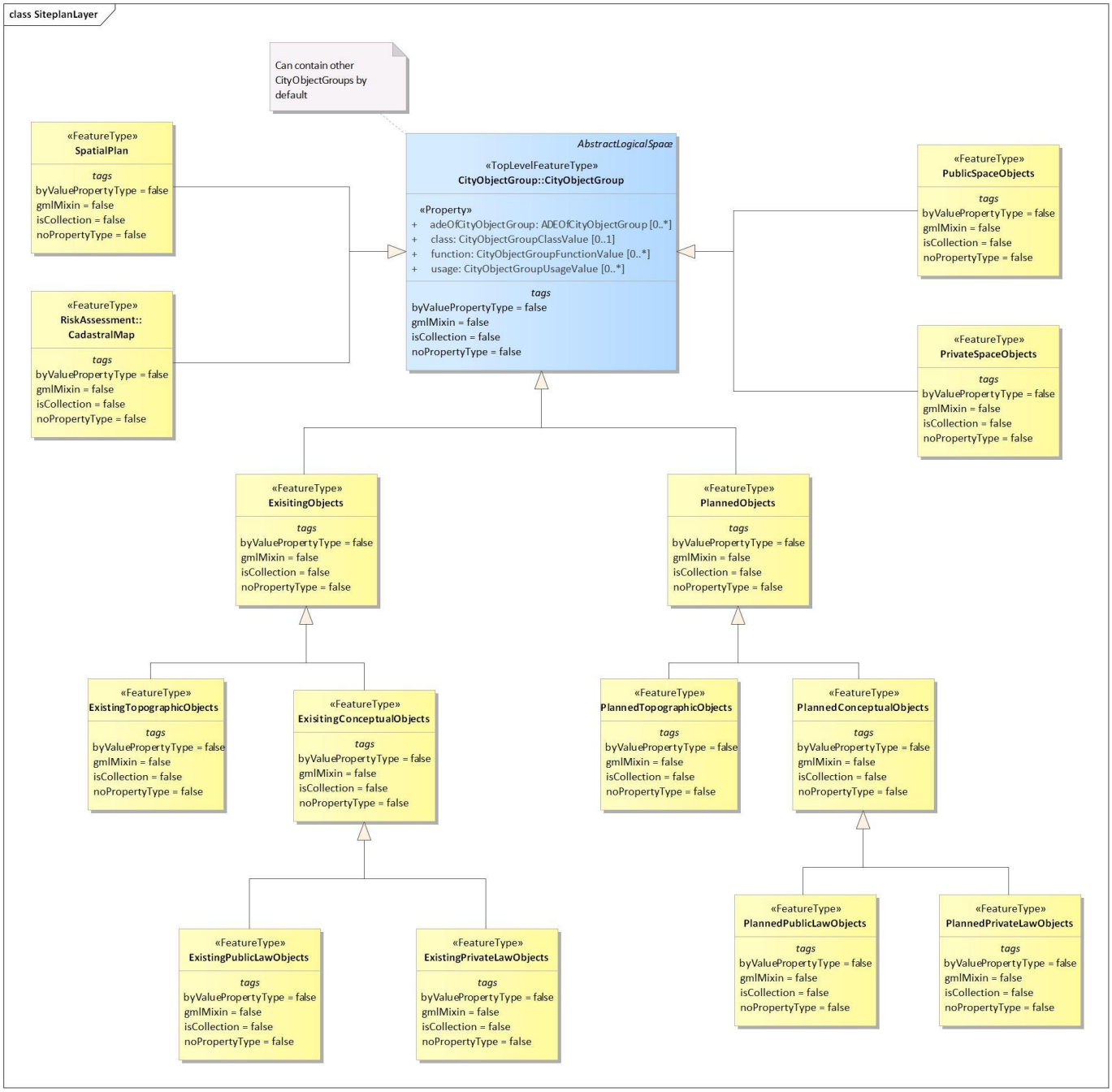


Diagram 38 - SiteplanLayer

1.15.1 ExisitingConceptualObjects

ExisitingConceptualObjects

Definition:	Grouping of existing objects which belong to the category of conceptual objects.
Subtype of:	ExisitingObjects
Supertype of:	ExistingPrivateLawObjects
	ExistingPublicLawObjects
Type:	Feature type

1.15.2 ExisitingObjects

ExisitingObjects	
Definition:	Grouping of existing objects which belong to both categories topographic objects and conceptual objects.
Subtype of:	CityObjectGroup
Supertype of:	ExisitingConceptualObjects
	ExistingTopographicObjects
Type:	Feature type

1.15.3 ExistingPrivateLawObjects

ExistingPrivateLawObjects	
Definition:	Grouping of existing objects of the category of private law.
Subtype of:	ExisitingConceptualObjects
Type:	Feature type

1.15.4 ExistingPublicLawObjects

ExistingPublicLawObjects	
Definition:	Grouping of existing objects of the category of public law.
Subtype of:	ExisitingConceptualObjects
Type:	Feature type

1.15.5 ExistingTopographicObjects

ExistingTopographicObjects	
Definition:	Grouping of existing objects which belong to the category of topographic objects.
Subtype of:	ExisitingObjects
Type:	Feature type

1.15.6 PlannedConceptualObjects

PlannedConceptualObjects

Definition:	Grouping of existing objects which belong to the category of conceptual objects.
Subtype of:	PlannedObjects
Supertype of:	PlannedPrivateLawObjects
	PlannedPublicLawObjects
Type:	Feature type

1.15.7 PlannedObjects

PlannedObjects

Definition:	Grouping of existing objects which belong to both categories topographic objects and conceptual objects.
Subtype of:	CityObjectGroup
Supertype of:	PlannedConceptualObjects
	PlannedTopographicObjects
Type:	Feature type

1.15.8 PlannedPrivateLawObjects

PlannedPrivateLawObjects

Definition:	Grouping of planned objects of the category of private law.
Subtype of:	PlannedConceptualObjects
Type:	Feature type

1.15.9 PlannedPublicLawObjects

PlannedPublicLawObjects

Definition:	Grouping of planned objects of the category of public law.
Subtype of:	PlannedConceptualObjects
Type:	Feature type

1.15.10 PlannedTopographicObjects

PlannedTopographicObjects

Definition:	Grouping of existing objects which belong to the category of topographic objects.
Subtype of:	PlannedObjects

Type:	Feature type

1.15.11 PrivateSpaceObjects

PrivateSpaceObjects	
Definition:	Grouping of objects which belong to the private space.
Subtype of:	CityObjectGroup
Type:	Feature type

1.15.12 PublicAuthority

PublicAuthority	
Definition:	The public authority validating the building permit application.
Subtype of:	BuildingPermitParty
Type:	Feature type

Association role	
Name:	lowerAuthority
Voidable:	false
Multiplicity:	0..*
Value type:	PublicAuthority (feature type)

1.15.13 PublicSpaceObjects

PublicSpaceObjects	
Definition:	Grouping of objects which belong to the public space.
Subtype of:	CityObjectGroup
Type:	Feature type

1.15.14 SpatialPlan

SpatialPlan	
Definition:	Grouping of city objects belonging to the spatial plan.
Subtype of:	CityObjectGroup
Type:	Feature type

D Python Dokumentation zum Algorithmus der Abstandsflächenberechnung

Clearance Space Computation

Felix Müller

Jan 24, 2021

CONTENTS:

1 CityGMLObject module 1

2 CityGMLProcessing module 9

3 ClearanceSpaceInspector module 15

4 DTM module 19

5 GeometricAnalysis module 21

6 RuleEngine module 39

7 WorldFileReader module 41

8 SampleCode.py 43

9 Indices and tables 51

Python Module Index 53

Index 55

CHAPTER

ONE

CITYGMLOBJECT MODULE

class CityGMLObject.**Building**

Bases: *CityGMLObject.CityGMLObject*

Class representing the CityGML class *Building*.

storeysAboveGround

Number defining the count of storeys above ground.

Type float

walls

List of *WallSurface* objects composing the building.

Type list(*WallSurface*)

roofs

List of *RoofSurface* objects composing the building.

Type list(*RoofSurface*)

buildingInstallations

List of *BuildingInstallation* objects composing the building.

Type list(*BuildingInstallation*)

outerCeilings

List of *OuterCeilingSurface* objects composing the building.

Type list(*OuterCeilingSurface*)

outerFloors

List of *OuterFloorSurface* objects composing the building.

Type list(*OuterFloorSurface*)

closureSurfaces

List of *ClosureSurface* objects composing the building.

Type list(*ClosureSurface*)

genericSurfaces

List of GenericThematicSurfaces objects composing the building.

Type list(GenericThematicSurfaces)

groundSurfaces

List of *GroundSurface* objects composing the building.

Type list(*GroundSurface*)

buildingParts
List of *BuildingPart* objects composing the building.

Type list(*BuildingPart*)

clearanceSpace
Clearance space of the building (union of the clearance surfaces of all building components).

Type *ClearanceSurface*

buildingClearanceSpace()
Computes and returns a union of all clearance spaces contained by any building components representing the clearance space of the building.

Returns Geometric union of all clearance spaces contained by the building components in the x y plane.

Return type shapely.geometry.polygon.Polygon

Note: Under particular configurations the return type might be of the type *shapely.geometry.multipolygon.MultiPolygon*.

---

getComponents()
Returns the list of all building components. Building installations are not unpacked.

Returns List of the contained *SurfacePartsContainer* objects.

Return type list(*SurfacePartsContainer*)

unpackedComponentsAsList()
Returns the list of all surface part containers of the building. Building installations are unpacked to their surface part containers.

Returns List of the contained *SurfacePartsContainer* objects.

Return type list(*SurfacePartsContainer*)

class CityGMLObject.**BuildingInstallation**(*identifier*)
Bases: *CityGMLObject.CityGMLObject*

Class representing the CityGML class *BuildingInstallation*.

surfaceContainers
List of *SurfacePartsContainer* objects composing the building installtion instance.

Type list(SurfacePartsContainers)

aggregatedBuildingInstallations
List of *BuildingInstallation* objects which are adjacent and secondary building installations.

Type list(*BuildingInstallation*)

getComponents()
Returns a list of *SurfacePartsContainer* objects composing the building installtion instance. The components of the aggregated building installations are included.

Returns List of the contained *SurfacePartsContainer* objects.

Return type list(*SurfacePartsContainer*)

getSurfaceParts()
Returns a list of the contained *BuildingSurfacePart* objects. Each *SurfacePartsContainer* object is unpacked to its surface parts.

Returns List of the contained *BuildingSurfacePart* objects.

Return type list(*BuildingSurfacePart*)

class CityGMLObject.**BuildingPart**
Bases: *CityGMLObject.Building*

Class representing the CityGML class *BuildingPart*.

class CityGMLObject.**BuildingSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.Surface*

Subclass of *Surface* for the representation of surface parts contained by a *SurfacePartsContainer*.

roofEdges
List of dictionaries holding a list of adjacent edges and a reference to the adjacent RoofSurface object.

Type list(dict("edges": list([tuple(x0, y0, z0), tuple(x1, y1, z1)]), "roof": RoofSurface))

clearanceSpaceSurfaces
List containing the resulting clearance space surfaces.

Type list(ClearanceSpaceSurface)

class CityGMLObject.**CityGMLObject**
Bases: object

Base class for the representation of CityGML objects in the context of a clearance space computation.

id
Unique identifier.

Type str

parent
Parent object for the hierarchical representation of relationships between CityGML objects.

Type *CityGMLObject*

adjacentObjects
List of spatially adjacent objects.

Type list(*SurfacePartsContainer*)

classCode
Representation of the CityGML class code.

Type str

functionCode
Representation of the CityGML function code.

Type str

usageCode
Representation of the CityGML usage code.

Type str

objectRelations
List of object relations between the CityGML objects.

Type list(*ObjectRelation*)

referenceChildsWithParent(*childs*)
Recursive method to reference the child objects with their parent objects.

Parameters **childs** (CityGMLObject) – List of the contained CityGMLObject instances provided by the parent object.

Returns

Return type None.

class CityGMLObject.**ClearanceSurface**(*identifier*, *polygon*)
Bases: *CityGMLObject.Surface*, *CityGMLObject.CityGMLObject*

Class representing the clearance surface of building components.

clearanceDepth
Depth of the clearance surface in meters.

Type float

computationDescription
Text describing the used equations to compute the depth of the clearance surface.

Type str

adjustmentFactorRoof
Value of the adjustment factor depending on the slope of adjacent roofs.

Type float

adjustmentFactorSpatialPlan
Value of the adjustment factor depending on the location in the spatial plan.

Type float

class CityGMLObject.**ClosureSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *ClosureSurface*.

class CityGMLObject.**ClosureSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *ClosureSurface*.

class CityGMLObject.**Door**(*identifier*)
Bases: *CityGMLObject.SurfacePartsContainer*

Class representing the CityGML thematic surface class *Door*.

class CityGMLObject.**DoorSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.Surface*

Class representing a surface part/member of the CityGML thematic surface class *Door*.

class CityGMLObject.**GenericThematicSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *GenericThematicSurface*.

class CityGMLObject.**GenericThematicSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *GenericThematicSurface*.

class CityGMLObject.**GroundSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *GroundSurface*.

class CityGMLObject.**GroundSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *GroundSurface*.

class CityGMLObject.**ObjectRelation**(*xLinkID*, *relationType*)
Bases: object

Class for the representation of object relations between CityGML objects.

objectID
Object identifier provided in the xlink:href attribute pointing to the reference element.

Type str

relationType
Code list value to specify the relation between the objects.

Type str

class CityGMLObject.**OuterCeilingSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *OuterCeilingSurface*.

class CityGMLObject.**OuterCeilingSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *OuterCeilingSurface*.

class CityGMLObject.**OuterFloorSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *OuterFloorSurface*.

class CityGMLObject.**OuterFloorSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *OuterFloorSurface*.

class CityGMLObject.**PropertyParcel**(*identifier*, *polygon*)
Bases: *CityGMLObject.Surface*

Class representing a single instance or aggregation of instances of the CityGML SiteplanADE class *AX_Flurstueck*.

class CityGMLObject.**RoofSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *RoofSurface*.

class CityGMLObject.**RoofSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *RoofSurface*.

class CityGMLObject.**SpatialPlan**(*developmentAreas*, *factorDictionary*)
Bases: object

Class representing the CityGML SiteplanADE object *SpatialPlan*.

developmentAreas
List of *SpatialPlanDevelopmentArea* objects contained by the spatial plan in which the building is located.

Type list(*SpatialPlanDevelopmentArea*)

developmentAreaFactors
Dictionary holding the clearance space adjustment factors. The format is e.g.:

```
spatialPlanFactors = {
    "öffentlicheVerkehrsfläche": 0.2,
    "öffentlicheWasserfläche": 0.2,
    "öffentlicheGrünfläche": 0.2,
    "Kleinsiedlungsgebiet": 0.4,
    "reinesWohngebiet": 0.4,
    "allgemeinesWohngebiet": 0.4,
    "besonderesWohngebiet": 0.4,
    "Dorfgebiet": 0.4,
    "Mischgebiet": 0.4,
    "urbanesGebiet": 0.4,
    "Kerngebiet": 0.4,
    "Gewerbegebiet": 0.2,
    "Industriegebiet": 0.2,
    "Sondergebiet": 0.4
    };
```

Type dict(str : float)

class CityGMLObject.**SpatialPlanDevelopmentArea**(*identifier*, *polygon*, *areaType*)
Bases: *CityGMLObject.Surface*

Class representing the CityGML SiteplanADE class *ZoningElement*, *SupplementaryRegulation* or *BP_Baugebietsteilflaeche*.

areaType
Text defining the type of the development area.

Type str

class CityGMLObject.**Surface**(*identifier*, *polygon*)
Bases: *CityGMLObject.CityGMLObject*

Subclass of *CityGMLObject* for the representation of thematic surface parts.

geometry
Planar polygon geometry defining the surface.

Type shapely.geometry.polygon.Polygon

class CityGMLObject.**SurfacePartsContainer**(*identifier*)
Bases: *CityGMLObject.CityGMLObject*

Class acting as container for instances of *BuildingSurfacePart*. The usageCode is defaulted to "relevantForClearanceSpace".

surfaceParts
List surface parts composing the thematic surface (which acts as a container/aggregator class).

Type list(*BuildingSurfacePart*)

clearanceSpaceSurfaces
List containing the resulting clearance space surfaces.

Type list(ClearanceSpaceSurface)

getSurfaceParts()
Returns a list of the contained *BuildingSurfacePart* objects.

Returns List of the contained *BuildingSurfacePart* objects.

Return type list(*BuildingSurfacePart*)

class CityGMLObject.**ThematicBuildingSurfacePartsContainer**(*identifier*)
Bases: *CityGMLObject.SurfacePartsContainer*

Subclass of *SurfacePartsContainer* for the representation of thematic surfaces of the CityGML standard acting as container of instances of *BuildingSurfacePart*.

doors
List of *Door* objects contained by the thematic surface.

Type list(*Door*)

windows
List of *Window* objects contained by the thematic surface.

Type list(*Window*)

class CityGMLObject.**WallSurface**(*identifier*)
Bases: *CityGMLObject.ThematicBuildingSurfacePartsContainer*

Class representing the CityGML thematic surface class *WallSurface*.

class CityGMLObject.**WallSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.BuildingSurfacePart*

Class representing a surface part/member of the CityGML thematic surface class *WallSurface*.

class CityGMLObject.**Window**(*identifier*)
Bases: *CityGMLObject.SurfacePartsContainer*

Class representing the CityGML thematic surface class *Window*.

class CityGMLObject.**WindowSurfacePart**(*identifier*, *polygon*)
Bases: *CityGMLObject.Surface*

Class representing a surface part/member of the CityGML thematic surface class *Window*.

CHAPTER

TWO

CITYGMLPROCESSING MODULE

CityGMLProcessing.**dissolveThematicObjects**(*thematicObjects*, *surfacePartsContainerConstructor*, *surfacePartConstructor*, *doorSurfaceDesignation='Door'*, *windowSurfaceDesignation='Window'*, *objectType='WallSurface'*)

Converts the specified CityGML objects from DOM elements to an internal representation and returns them in a list.

Parameters

- **thematicObjects** (*xml.dom.minicompat.NodeList*) – List of DOM elements of the type specified in `objectType` that will be reconstructed.
- **surfacePartsContainerConstructor** (*class*) – Constructor of *CityGMLObject.SurfacePartsContainer* or any subclass of *CityGMLObject.SurfacePartsContainer*.
- **surfacePartConstructor** (*class*) – Constructor of *CityGMLObject.BuildingSurfacePart* or any subclass of *CityGMLObject.BuildingSurfacePart*.
- **doorSurfaceDesignation** (*str, optional*) – Due to the difference between the spatial concept between CityGML 2.0 and CityGML 3.0 also the desgination of door surfaces differ between the CityGML versions. The default is doorWindowTagNames["doorSurfaceTagName"].
- **windowSurfaceDesignation** (*str, optional*) – Due to the difference between the spatial concept between CityGML 2.0 and CityGML 3.0 also the desgination of door surfaces differ between the CityGML versions. The default is doorWindowTagNames["windowSurfaceTagName"].
- **objectType** (*str, optional*) – Name of the object type (according to the CityGML classes) that is currently processed. The default is "WallSurface".

Returns **objects** – List of reconstructed CityGML objects.

Return type list(*CityGMLObject.SurfacePartsContainer*)

CityGMLProcessing.**extractCityObjectRelations**(*relatedToElements*)

Extracts the xlink:href attributes and relation types from the DOM elements and returns them as a list of CityGMLObject.ObjectRelation objects.

Parameters **relatedToElements** (*list(xml.dom.minidom.Element)*) – List of DOM elements of the type <core:relatedTo>.

Returns **objectRelations** – List of object relations.

Return type list(*CityGMLObject.ObjectRelation*)

CityGMLProcessing.**extractCoords**(*geometryElement*)

Extracts the positions of the GML geometry.

Parameters **geometryElement** (*xml.dom.minidom.Element*) – GML DOM element holding the geometry properties.

Returns List of coordinate tuples.

Return type list(tuple(x, y, z))

CityGMLProcessing.**extractElement**(*parentElement*, *element='Building'*)

Extracts the building model from the CityGML instance document and instantiates a internal building representation.

Parameters

- **parentElement** (*xml.dom.minidom.Element*) – Parent element containing the wanted element.
- **element** (*str, optional*) – Name of the element to extract from parentElement. The default is "Building".

Returns Internal representation of the building of the CityGML document.

Return type *CityGMLObject.Building*

CityGMLProcessing.**extractGeometry**(*element*)

Extracts and returns the geometry of the DOM Element element which is assumed to be of the type <gml:surfaceMember> holding exactly ONE polygon geometry.

Parameters **element** (*xml.dom.minidom.Element*) – DOM element containing GML geometry.

Returns

- **listOfPositionsExterior** (*list(tuple(x, y, z))*) – List of the positions of the exterior ring of the geometry (polygon) as list of coordinate tuples.
- **listOfInteriorRings** (*list(list(tuple(x, y, z)))*) – List of the interior rings of the geometry (polygon). Each interior ring is a list of coordinate tuples.
- *polygonID* – The gml:id of the polygon geometry.

CityGMLProcessing.**extractPosFromPosElements**(*posElList*)

Extracts and returns the positions from a list of <gml:pos> elements.

Parameters **posElList** (*xml.dom.minicompat.NodeList*) – List of DOM elements of the type <gml:pos>.

Returns List of coordinate tuples.

Return type list(tuple(x, y, z))

CityGMLProcessing.**extractPosFromPosListElement**(*posListElement*, *srsDimension=3*, *tupleSeparator=' '*, *coordinateSeparator=' '*)

Exrtracts and returns the coordinates from the <gml:posList> posListElement.

Parameters

- **posListElement** (*xml.dom.minidom.Element*) – <gml:posList> element containing the coordinates.
- **srsDimension** (*int, optional*) – Dimension of the coordinate system. The default is 3.

- **tupleSeparator** (*str, optional*) – Separator used to distinguish the coordinate tuples. The default is ” “.
- **coordinateSeparator** (*str, optional*) – Separator used to distinguish the coordinate within the tuple. The default is ” “.

Returns List of coordinate tuples.

Return type list(tuple(x, y, z))

CityGMLProcessing.**extractSurfaceMembers**(*surfaceMembers*, *memberList*)

Recursive function storing surface member instances which are contained or referenced surfaceMember in the global array memberList.

Parameters

- **surfaceMembers** (*xml.dom.minicompat.NodeList*) – List of DOM elements of the type <gml:surfaceMember>.
- **memberList** (*list*) – Reference to global list to store the surface members in it.

Returns

Return type None.

CityGMLProcessing.**extractThematicSurfaces**(*parentElement*, *namespace*, *elementName*)

Extracts the thematic CityGML classes from parentElement and returns a list for of each of the thematic subclasses of ThematicSurfacePartsContainer.

Parameters

- **parentElement** (*xml.dom.minidom.Element*) – Parent element containing the thematic DOM elements from the CityGML document.
- **namespace** (*str*) – XML namespace which should be used to extract the elements matching the local node name of elementName.
- **elementName** (*str*) – Local node name neglecting the namespace prefix.

Returns List of nodes matching namespace and elementName.

Return type xml.dom.minicompat.NodeList

CityGMLProcessing.**fillingSurfaceTagNamesCityGML3 = {'doorSurfaceTagName': 'DoorSurface',**

Dictionary containing the designations for door and window surfaces of CityGML 3.0. It may be set as the global doorWindowTagNames by calling *setNamespacesAndCityGMLVersion()*.

CityGMLProcessing.**findElementInChilds**(*children*, *localChildName*, *number=None*, *namespace='*'*)

Returns the first *n* childs. *n* is specified by ‘number’. ‘localChildName’ is the node name without namespace prefix. By default number is None, which means all children will be extracted

Parameters

- **children** (*xml.dom.minicompat.NodeList*) – A DOM element’s child nodes.
- **localChildName** (*str*) – The node name without namespace prefix specifying the node type to search for.
- **number** (*int, optional*) – Number of child nodes to return. The default is None, which means all children matching the local name localChildName and XML namespace namespace.
- **namespace** (*str, optional*) – Defines the namespace the nodes, which are searched for, belong to. The default is “*”, which means all namespace are considered.

Returns List of *n* elements meeting the conditions of the input parameters.

Return type list(xml.dom.minidom.Element)

CityGMLProcessing.**getDescriptionFromDictionary**(*url*, *codeValue*)

CityGMLProcessing.**getElementChildNodes**(*element*)
Returns only child nodes of the type xml.dom.minidom.Element.

Parameters **element** (*xml.dom.minidom.Element*) – Element from which the child nodes should be extracted.

Returns List of child nodes of the type xml.dom.minidom.Element.

Return type list(xml.dom.minidom.Element)

CityGMLProcessing.**getFirstOfAType**(*childsOfCurrentNode*, *namespace*, *nodeName*)
Returns the first element that matches the nodeName from the list childsOfCurrentNode.

Parameters

- **childsOfCurrentNode** (*xml.dom.minicompat.NodeList*) – List of child nodes.
- **namespace** (*str*) – XML namespace to match the node name.
- **nodeName** (*str*) – Local node name without namespace prefix.

Returns The first node that matches the namespace and local node name.

Return type xml.dom.minidom.Node

CityGMLProcessing.**getThematicBuildingSurfacePartsContainers**(*parentElement*, *namespace='*'*)
Extracts the thematic CityGML classes from parentElement and returns a list for of each of the thematic subclasses of ThematicSurfacePartsContainer.

Parameters

- **parentElement** (*xml.dom.minidom.Element*) – Parent element containing the thematic DOM elements from the CityGML document.
- **namespace** (*str, optional*) – Defines the namespace of the CityGML elements. The default is '*'.

Returns

- **wallSurfaces** (*list(CityGMLObject.WallSurface)*) – List of *CityGMLObject.WallSurface* objects.
- **roofSurfaces** (*list(CityGMLObject.RoofSurface)*) – List of *CityGMLObject.RoofSurface* objects.
- **groundSurfaces** (*list(CityGMLObject.GroundSurface)*) – List of *CityGMLObject.GroundSurface* objects.
- **outerFloors** (*list(CityGMLObject.OuterFloorSurface)*) – List of *CityGMLObject.OuterFloorSurface* objects.
- **outerCeilings** (*list(CityGMLObject.OuterCeilingSurface)*) – List of *CityGMLObject.OuterCeilingSurface* objects.
- **closureSurfaces** (*list(CityGMLObject.ClosureSurface)*) – List of *CityGMLObject.ClosureSurface* objects.
- **genericThematicSurfaces** (*list(CityGMLObject.GenericThematicSurface)*) – List of *CityGMLObject.GenericThematicSurface* objects.

CityGMLProcessing.**hasOrientableSurface**(*member*)
Returns true if member contains a <gml:OrientableSurface> element.

Parameters **member** (*xml.dom.minidom.Element*) – DOM element to inspect.

Returns Returns true if a descendant is of the type <gml:OrientableSurface>.

Return type bool

CityGMLProcessing.**openingTagNamesCityGML2 = {'doorSurfaceTagName': 'Door', 'windowSurface'**
Dictionary containing the designations for door and window surfaces of CityGML 2.0. It may be set as the global doorWindowTagNames by calling *setNamespacesAndCityGMLVersion()*.

CityGMLProcessing.**parseCoordinates**(*text*, *separator=' '*)
Extracts coordinates from a text string by the use of the specified separator and additionally transforms them with the provided transformation parameters *transformationParams*.

Parameters

- **text** (*str*) – Text containing the coordinates.
- **separator** (*str, optional*) – Separator to divide text in the single coordinate values. The default is " ".

Returns

- **x_trans** (*float*) – Parsed and transformed x coordinate.
- **y_trans** (*float*) – Parsed and transformed y coordinate.
- **z_trans** (*float*) – Parsed and transformed z coordinate.

CityGMLProcessing.**removeChildsFromParent**(*elementList*)
Removes each element in elementList from its parent node and the DOM.

Parameters **elementList** (*xml.dom.minicompat.NodeList*) – List of DOM elements to remove from the DOM.

Returns

Return type None.

CityGMLProcessing.**resolveCodeListElement**(*codeListElements*)
Returns the description for the code value provided in the first entry of codeListElements.

Parameters **codeListElements** (*list(xml.dom.minidom.Element)*) – List of DOM elements holding code list values.

Returns **codeDescription** – Description for the corresponding code list value.

Return type str

CityGMLProcessing.**setNamespacesAndCityGMLVersion**(*domObject*)
Sets the required namespaces and CityGML version for the extraction from the CityGML document.

Note: It is necessary to have an inline profile within the instance document for the function to work.

Parameters **domObject** (*xml.dom.minidom.Document*) – Document object of the instance document of the building model.

Returns

Return type None.

CityGMLProcessing.**solveXLink**(*e*, *element_id*)

Returns the element with `element_id` that is referenced by `e`'s xlink:href atrtibute.

Parameters

- **e** (*xml.dom.minidom.Element*) – Element that holds the xlink:href attribute referencing an element.
- **element_id** (*str*) – XPointer syntax to reference the element via the gml:id attribute.

Returns The element that was referenced by the xlink:href atrtibute in `e`.

Return type xml.dom.minidom.Element

CityGMLProcessing.**transformationParams = [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]**

Transformation parameters for the geo-referencing matrix. The the building coordinates are transformed from the local CRS (coordinate reference system) to the global CRS as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & 0 & C \\ D & E & 0 & F \\ 0 & 0 & 1 & G \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The *transformationParams* stores the parameters as follows: [A, B, C, D, E, F, G]

CHAPTER

THREE

CLEARANCESPACEINSPECTOR MODULE

class ClearanceSpaceInspector.**ClearanceSpaceInspector**(*spatialPlan*, *propertyParcel*, *building*, *digitalTerrainModel*, *buildingTICs*)

Bases: `object`

Inspector class to conduct a clearance space analysis on a building.

spatialPlan
The spatial plan in whose development areas the inspected building is located.

Type *CityGMLObject.SpatialPlan*

propertyParcel
The property parcel on which the building is/will be errected.

Type *CityGMLObject.PropertyParcel*

building
The building which should be inspected on clearance spaces.

Type *CityGMLObject.Building*

digitalTerrainModel
The digital terrain model in the near surrounding of the building as coordinate tuples of the grid points with the corresponding height values.

Type tuple(list([x, y]), list(z))

buildingTICs
The terrain intersection curves of the building.

Type list(shapely.geometry.linestring.LineString)

__setClearanceSpaceDepth(*rule='(buildingClass == "buildingClass1" or buildingClass == "buildingClass2") and storeysAboveGround <= 3'*, *value=3.0*)
Evaluates `rule` and sets `value` as the global clearance space depth if the rule evaluation yields a positive result (=true).

Parameters

- **rule** (*str, optional*) – A rule defined by law privileging certain buildings with a default clearance space depth. The default is '(buildingClass == "buildingClass1" or buildingClass == "buildingClass2") and storeysAboveGround <= 3'.
- **value** (*float, optional*) – Value for the default clearance space depth in meters. The default is 3.0.

Returns

Return type None.

__setSpatialPlanAdjustmentFactor()
Sets the clearance space adjustment factor based on the development area in which the building is located.

Returns

Return type None.

calculateClearanceSpace(*component*)
Function that is called for the clearance space computation if a clearance space for a building component is required.

Parameters **component** (CityGMLObject.ThematicBuildingSurfacePartsContainer) – Component for which the clearance spaces will be computed.

Returns

Return type None.

clearanceSpace(*surfacePart*, *buildingTICs*, *digitalTerrainModel*, *depthAdjustmentFactor=0.4*, *clearanceSpaceDepth=None*, *minimumClearanceSpaceDepth=3.0*)
Method computing the clearance space surface for the surface part of a building component. The clearance space surface computation is conducted for each line segment of the surface part.

Parameters

- **surfacePart** (CityGMLObject.BuildingSurfacePart) – Surface part requiring a clearance space surface.
- **buildingTICs** (*list(shapely.geometry.linestring.LineString)*) – The terrain intersection curves of the building.
- **digitalTerrainModel** (*tuple(list([x, y]), list(z))*) – The digital terrain model in the near surrounding of the building as coordinate tuples of the grid points with the corresponding height values.
- **depthAdjustmentFactor** (*float, optional*) – Factor applied on the computed clearance space depth. The default is 0.4.
- **clearanceSpaceDepth** (*float, optional*) – Universal clearance space depth, which results from the evaluation in *__setClearanceSpaceDepth*. The default is None.
- **minimumClearanceSpaceDepth** (*float, optional*) – Lower threshold of the measure of a clearance space depth in meters. The default is 3.0.

Returns

Return type None.

inspectBuilding4ClearanceSpaces()
Starts the inspection on clearance spaces.

Returns

Return type None.

privilegeComponents()
Inspects the building's components on privileged components which will be excluded from the clearance space computation.

Returns

Return type None.

recursiveInspection(*obj*)
Recursive method deciding for each object component if there will be a clearance space computation.

Parameters **obj** (CityGMLObject.CityGMLObject) – Current object whose components will be examined on requirement of a clearance space.

Returns Required due to the recursive call of recursiveInspection so that there is not called the clearance space calculation on the building instance or building installation instance.

Return type bool

setClearanceSpaceConfiguration(*rule=None*)

Sets the rule which will be forwarded to the private method *__setClearanceSpaceDepth()*. Calls the private *__setSpatialPlanAdjustmentFactor()* method.

Parameters **rule** (*str, optional*) – A rule defining the constraints for a general clearance space depth. The default is None.

Returns

Return type None.

setConsequenceFunction(*func*)

Method to set the function which will be conducted if a clearance space is required.

Parameters **func** (*function*) – Function which will conduct the clearance space computation.

Returns

Return type None.

setDefaultAdjustmentFactor(*value=0.4*)

Sets the default adjustment factor for the clearance space depth.

Parameters **value** (*float, optional*) – Default adjustment factor which will be multplied with the clearance space depth. The default is 0.4.

Returns

Return type None.

setMinimumClearanceSpaceDepth(*value=3.0*)

Sets the minimum threshold of the clearance space depth.

Parameters **value** (*float, optional*) – Minimum threshold in meters concerning the clearance space depth. The default is 3.0.

Returns

Return type None.

setRuleEngine(*ruleEngine*)

Method to set the clearance space inspector's rule engine.

Parameters **ruleEngine** (RuleEngine.RuleEngine) – Rule engine examining a decision tree which defines the constraints which must be met to make a clearance space obligatory.

Returns

Return type None.

CHAPTER

FOUR

DTM MODULE

DTM.**createTIN**(*points*, *clippingObject=None*)
Creates a triangulated irregular network (TIN) from the input points and cuts out the clipping geometry. The clipping is conducted in 2D.

Parameters

- **points** (*list(shapely.geometry.point.Point)*) – Points which are used to generate the TIN.
- **clippingObject** (*shapely.geometry.polygon.Polygon, optional*) – Polygon which is used to cut out areas from the TIN. The default is None.

Returns List of polygons (triangles) defining the TIN.

Return type list(shapely.geometry.polygon.Polygon)

DTM.**dtmPointsAsGridPoints**(*dtmPoints*)
Separates 3D points in 2D positions and z values.

Parameters **dtmPoints** (*list(shapely.geometry.point.Point)*) – 3D points.

Returns

- *list([x, y])* – List of 2D positions of the input points.
- *list(z)* – List of z value of the input points.

DTM.**readDTM**(*filename*, *clippingPolygon*)
Reads a DTM from a ASCII-XYZ file and returns points inside of the clipping geometry `clippingPolygon`. The spatial filtering is conducted only in 2D (1st and 2nd coordinate components).

Parameters

- **filename** (*str*) – Path to the ASCII xyz.-file. The file should not contain a heading with column designation.
- **clippingPolygon** (*shapely.geometry.polygon.Polygon*) – Polygon geometry which is used to only filter points which are inside of the polygon.

Returns List of points filtered by the clipping polygon.

Return type list(shapely.geometry.point.Point)

DTM.**readDTMAsDataPoints**(*filename*, *clippingPolygon*)
Reads a DTM from a ASCII-XYZ file and returns points inside of the clipping geometry `clippingPolygon`. The spatial filtering is conducted only in 2D (1st and 2nd coordinate components).

Parameters

- **filename** (*str*) – Path to the ASCII xyz.-file. The file should not contain a heading with column designation.
- **clippingPolygon** (*shapely.geometry.polygon.Polygon*) – Polygon geometry which is used to only filter points which lay inside of the polygon.

Returns

- *numpy.ndarray* – 2D array containing the x (1st column) y (2nd column) positions of the DTM.
- *numpy.ndarray* – Array containing the z values to the corresponding data positions.

DTM.**render**(*axis*, *polygons*)

Renders a TIN.

Parameters

- **axis** (*matplotlib.axes._subplots.Axes3DSubplot*) – Axis used for rendering.
- **polygons** (*list(shapel.geometry.polygon.Polygon)*) – TIN triangles.

Returns

Return type None.

CHAPTER

FIVE

GEOMETRICANALYSIS MODULE

GeometricAnalysis.**aggregateBuildingInstallations**(*buildingInstallations*)

Aggregates adjacent building installations if they are of different order, i.e. a primary building installation aggregated a secondary building installation.

Parameters **buildingInstallations** (*list(CityGMLObject.BuildingInstallation)*) – List of the building installation objects.

Returns

Return type None.

GeometricAnalysis.**aggregateSurfaces**(*surfaces*)

Conducts a union operation in the x y plane of the surfaces provided as polygons. Only considers polygons with non-zero area.

Parameters **surfaces** (*list(shapely.geometry.polygon.Polygon)*) – List of polygon geometries.

Returns **u** – Union of all polygons with non-zero area.

Return type shapely.geometry.polygon.Polygon

GeometricAnalysis.**checkAdjacency**(*object0*, *object1*, *buffer=0.05*)

Checks if there is an adjacency between `object0` and `object1`.

Parameters

- **object0** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Object for which the the adjacency is checked.
- **object1** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Candidate object to be checked if it is adjacent to `object0`.
- **buffer** (*float, optional*) – Tolerance value in meters. The default is 0.05.

Returns Adjacent surface parts belonging to `object1`.

Return type list(*CityGMLObject.BuildingSurfacePart*)

GeometricAnalysis.**checkEdgeAdjacency**(*polygonA*, *polygonB*, *distanceBuffer=0.5*, *toleranceForAngle=1*)

Conducts an adjacency analysis between two polygons on line segment level.

Parameters

- **polygonA** (*list(tuple(x, y [, z]))*) – List of coordinate tuples of describing the exterior ring of the polygon.

- **polygonB** (*list(tuple(x, y [, z]))*) – List of coordinate tuples of describing the exterior ring of the polygon.
- **distanceBuffer** (*float, optional*) – Tolerance value for edge distance in meters. The default is 0.5.
- **toleranceForAngle** (*TYPE, optional*) – Tolerance value for edge parallelity in degrees. The default is 1.

Returns **adjacentEdges** – List of lists containing the coodinate tuples of the edge.

Return type list([tuple(x, y [, z]), tuple(x, y [, z])])

GeometricAnalysis.**component2PointCollection**(*obj*)

Extracts the points from all geometries contained by `obj` and returns them as a list.

Parameters **obj** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Object to extract the point from its geometries.

Returns List of points contained by `obj`.

Return type list(numpy.ndarray)

GeometricAnalysis.**componentSlope**(*component*)

Returns the mean slope of the component where each surface part is weighted by its contributing area.

Parameters **component** (CityGMLObject.SurfacePartsContainer) – Instance of :py:class`CityGMLObject.SurfacePartsContainer` or of any subclass of :py:class`CityGMLObject.SurfacePartsContainer`.

Returns **slopeAngle** – Mean slope angle in degrees.

Return type float

GeometricAnalysis.**computeHeightAdjustment**(*surfacePart*, *edge*)

Returns the clearance space depth adjustment of adjacent roofs, i.e the height of the roof multiplied with the height adjustment factor for adjacent roofs.

Parameters

- **surfacePart** (CityGMLObject.BuildingSurfacePart) – Inspected surface part where `edge` is part of requiring a clearance space surface.
- **edge** (*list(tuple(x, y, z)) with length = 2*) – List of two coordinate tuples defining an edge.

Returns **adjustment** – Bias from the adjacent roof which will be added to clearance space depth.

Return type float

GeometricAnalysis.**computeLineIntersection3D**(*pA*, *vecA*, *pB*, *vecB*, *epsilon=1e-05*)

Computes the intersection point between two lines. If it exists it returns a tuple containing the intersection point, the type of intersection as a string and the scale factors in `x`, where `x[0]` is the scale factor for `vA` and `x[1]` is the scale factor for `vB`.

Parameters

- **pA** (*numpy.ndarray*) – Start point of lineA.
- **vecA** (*numpy.ndarray*) – Direction vector from start point to end point of lineA.
- **pB** (*numpy.ndarray*) – Start point of lineB.
- **vecB** (*numpy.ndarray*) – Direction vector from start point to end point of lineB.
- **epsilon** (*float, optional*) – Error tolerance in meters. The default is 10**-5.

Returns

- *numpy.ndarray* – Intersection point.
- *str* – Type of intersection in text form.
- **x** (*numpy.ndarray*) – Scale factors in `x`, where `x[0]` is the scale factor for `vA` and `x[1]` is the scale factor for `vB`.

`GeometricAnalysis.`**`createDTMBuffer`**(*buildingComponents*, *bufVal=4.0*)

Creates a 2D buffer around the building to extract points from the digital terrain model (DTM).

Parameters

- **`buildingComponents`** (*list(*`CityGMLObject.SurfacePartsContainer`*)*) – List of the building components.
- **`bufVal`** (*`float, optional`*) – Buffer value in meters. The default is 4.0.

Returns

- **buildingXYProjection** (*shapely.geometry.polygon.Polygon*) – A 2D union (polygon) of the building components projected on the x y plane.
- **buildingDTMBuffer** (*shapely.geometry.polygon.Polygon*) – Buffered `buildingXYProjection` by the value of `bufVal`.

`GeometricAnalysis.`**`createTopologies`**(*building*)

Analyses the building components on spatial adjacency to other building components and adds the references of adjacent objects to `CityGMLObject.CityObject.adjacentObjects` resulting in a graph structure of the builiding.

Parameters **`building`** (`CityGMLObject.Building`) – Building object.

Returns

Return type None.

`GeometricAnalysis.`**`distanceAlongAxis_Component`**(*obj*, *vector*)

Returns the length of `obj` projected on the axis defined by `vector`. Vector should be normalized. Polygon is a shapely geometry.

Parameters

- **`obj`** (`CityGMLObject.SurfacePartsContainer` *or* `CityGMLObject.BuildingInstallation`) – Object whose length is computed.
- **`vector`** (*`numpy.ndarray`*) – Normalized projection axis vector.

Returns **dmax** – Length of component projected along the projection axis `vector`.

Return type float

`GeometricAnalysis.`**`distanceAlongAxis_SurfacePart`**(*polygon*, *vector*)

Returns a list of lengths of `polygon` projected on the axis defined by `vector`. Vector should be normalized. Polygon is a shapely geometry.

Parameters

- **`polygon`** (*`shapely.geometry.polygon.Polygon`*) – Polygon object.
- **`vector`** (*`numpy.ndarray`*) – Normalized projection axis vector.

Returns **projectedDistances** – List of lengths of `polygon` along the projection axis defined by `vector`.

Return type list(float)

GeometricAnalysis.**distanceAlongProjectionAxis**(*refP*, *dirVec*, *p*)

Returns the distance between the reference point `refP` and point `p` projected on the projection axis defined by `dirVec`.

Parameters

- **refP** (*list(float)*) – List containing the coordinates of the reference point.
- **dirVec** (*numpy.ndarray*) – Projection axis vector.
- **p** (*numpy.ndarray*) – List containing the coordinates of the point.

Returns **d** – The projected distance.

Return type float

GeometricAnalysis.**distanceBetweenLines**(*pointA*, *vecA*, *pointB*)

Copmutes the distances between to lines.

Parameters

- **pointA** (*numpy.ndarray*) – Point on lineA.
- **vecA** (*numpy.ndarray*) – Not normalized direction vector of the lineA.
- **pointB** (*numpy.ndarray*) – Point on lineB.

Returns Distance between the to lines.

Return type float

GeometricAnalysis.**distancePoint2Plane**(*normalVec*, *referencePlanePoint*, *point*)

Copmutes the distance of `point` to the plane defined by the plane normal `normalVec` and the plane point `refrencePlanePoint`.

Parameters

- **normalVec** (*numpy.ndarray*) – Not normalized plane normal vector.
- **referencePlanePoint** (*tuple(x, y, z)*) – Coordinate tuple.
- **point** (*tuple(x, y, z)*) – Coordinate tuple.

Returns Returns the signed distance from `point` to the plane.

Return type float

GeometricAnalysis.**extractGeometriesFromComponent**(*obj*)

Returns a list of geometries contained by `obj`.

Parameters **obj** (CityGMLObject.SurfacePartsContainer) – Instance of :py:class`CityGMLObject.SurfacePartsContainer` or of any subclass of :py:class`CityGMLObject.SurfacePartsContainer`.

Returns **geometries** – List of shapely polygons.

Return type list(shapely.geometry.polygon.Polygon)

GeometricAnalysis.**findAdjacentRoofPartsAbove**(*componentSurfacePart*, *angleTolerance=2.0*)

Finds the roofs from the list of adjacent roofs which are relevant for `componentSurfacePart`, i.e. finds roofs which are at least partly above the component.

Parameters

- **componentSurfacePart** (CityGMLObject.BuildingSurfacePart) – Instance of :py:class`CityGMLObject.BuildingSurfacePart` or of any subclass of :py:class`CityGMLObject.BuildingSurfacePart`.

- **angleTolerance** (*float, optional*) – Tolerance value in degrees. The default is 2.0.

Returns

Return type None.

GeometricAnalysis.**findClosestDistance2PropertyBoundary**(*obj*, *propertyParcel*)

Returns the minimum distance of a component obj to the property boundary propertyParcel. The component is assumed to be located inside of the property boundary.

Parameters

- **obj** (CityGMLObject.SurfacePartsContainer) – The building component that is analyzed.
- **propertyParcel** (CityGMLObject.PropertyParcel) – The property parcel of the building.

Returns The closeest distance between obj and the property boundary.

Return type float

GeometricAnalysis.**findPlaneNormal**(*polygon*)

Returns the polygon plane normal vector.

Parameters **polygon** (*list(tuple(x, y, z))*) – List of coordinate tuples describing the exterior ring of the polygon.

Returns **n** – Polygon face normal vector not normalized.

Return type numpy.ndarray

GeometricAnalysis.**findPropertyBoundarySegments_ONRelation**(*component*, *propertyParcel*, *propertyBoundaryBuffer=0.1*)

Returns the line segment/s which are touched by the component/building projected on the x y plane. It is assumed that the building is inside of the property boundary and at most is touching a part of the boundary.

Parameters

- **component** (CityGMLObject.SurfacePartsContainer) – Building component containing surface parts that is inspected on topolgical relations w.r.t. propertyBoundary.
- **propertyParcel** (CityGMLObject.PropertyParcel) – Property parcel of the building.
- **propertyBoundaryBuffer** (*TYPE, optional*) – Tolerance value in meters. The default is 10**-1.

Returns List of property border line segments, each stored as list of coordinate tuples.

Return type list(list(tuple(x, y)))

GeometricAnalysis.**genericCheck**(*obj*, *checkingFunction*)

Generic checking function to return the function value of checkingFunction.

Parameters

- **obj** (*any*) – Object of any type.
- **checkingFunction** (*function*) – Function that takes as input parameter the provided obj.

Returns Returns the function value of checkingFunction. The return value can be of any type.

Return type Any

GeometricAnalysis.**genericExceedsThresholds**(*checkingFunctions*, *obj*, *negativeConsequenceFunc=None*, *positiveConsequenceFunc=None*)

Conducts tests on component which are provided as list of testing functions. If any function returns true the function returns true (i.e. it is a logical 'or' expression).

Parameters

- **checkingFunctions** (*list(functions)*) – Functions applied on obj returning a boolean value as result.
- **obj** (*object*) – Object that is the parameter for the functions in checkingFunctions.
- **negativeConsequenceFunc** (*funnction, optional*) – Function that is applied on obj if the checking result is negative (=false). The default is None.
- **positiveConsequenceFunc** (*TYPE, optional*) – Function that is applied on obj if the checking result is positive (=true). The default is None.

Returns Returns true if any of the checking functions returns true. Returns false otherwise.

Return type bool

GeometricAnalysis.**geometricOperation**(*obj*, *objFunction*, *otherObj*)

Performs the function :py:attr`objFunction` called by obj with otherObject as function parameter.

Parameters

- **obj** (*shapely.geometry*) – Geometry object.
- **objFunction** (*function*) – A function that is called by obj.
- **otherObj** (*shapely.geometry*) – Object as input argument for :py:attr`objFunction`.

Returns Returns the function return value, which can be of any type.

Return type Any

GeometricAnalysis.**getClearanceSpaceDepthAdjustmentFactor**(*surfacePartLineSegment*, *clearanceSpaceDirection*, *devAreaOfBuilding*, *propertyParcel*, *spatialPlan*, *defaultAdjustmentFactor=0.4*, *publicSpacesRule='targetType in ["öffentlicheVerkehrsfläche", "öffentlicheGrünfläche", "öffentlicheWasserfläche"]'*, *developmentAreaRule='areaType in ["urbanesGebiet", "Kerngebiet"]'*)

Returns the adjustment factor for the computation of the clearacen space depth based on the spatial configuration w.r.t. the public spaces and adjacent development areas.

Parameters

- **surfacePartLineSegment** (*list(numpy.ndarray) with length = 2*) – Inspected line segment of a surface part that rewuires a clearance space surface.

- **clearanceSpaceDirection** (*numpy.ndarray*) – Normalized direction vector of the extent of the clearance space surface, i.e. the plane normal of the surface part.
- **devAreaOfBuilding** (*str*) – Development area type of the building.
- **propertyParcel** (CityGMLObject.PropertyParcel) – Property parcel of the building.
- **spatialPlan** (CityGMLObject.SpatialPlan) – Spatial plan for the region of the building.
- **defaultAdjustmentFactor** (*float, optional*) – Default adjustment factor. The default is 0.4.
- **publicSpacesRule** (*str, optional*) – A rule against which the building's spatial configuration will be checked w.r.t. public spaces. The default is 'targetType in ["öffentlicheVerkehrsfläche", "öffentlicheGrünfläche", "öffentlicheWasserfläche"]'.
- **developmentAreaRule** (*TYPE, optional*) – A rule against which the building's spatial configuration will be checked w.r.t. development areas. The default is 'areaType in ["urbanesGebiet", "Kerngebiet"]'.

Returns **depthAdjustmentFactor** – The appropriate adjustment factor which yield the rules of the spatial configuration of the building.

Return type float

GeometricAnalysis.**getComponentByIDRelationAndType**(*element*, *relationType*, *objectTypes=<class 'object'>*)

Returns the components which are referenced by the *CityGMLObject.ObjectRelation* instances in `element.objectRelations` with the specified relation types `relationType` and of types contained in `objectTypes`.

Parameters

- **element** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Object from which the related objects are extracted and returned.
- **relationType** (*str*) – Code list value describing the relation type.
- **objectTypes** (*tuple(class0, class1, ..), optional*) – Tuple of types/classes to search for in `element.objectRelations`. The default is (object).

Returns **components** – List of components matching the relation type `relationType` and object types defined in `objectTypes`.

Return type list(*CityGMLObject.SurfacePartsContainer* or *CityGMLObject.BuildingInstallation*)

GeometricAnalysis.**getDevelopmentAreaTypeOfBuilding**(*building*, *spatialplan*)

Returns the type of development area in which `building` is located.

Parameters

- **building** (CityGMLObject.Building) – Building object.
- **spatialplan** (CityGMLObject.SpatialPlan) – Spatial plan object.

Returns Returns *CityGMLObject.SpatialPlanDevelopmentArea.areaType* of the development area in which `building` is located.

Return type str

GeometricAnalysis.**getLengthProjectionVectorsOfAdjacentObjects**(*adjacentObjects*)
Returns length projection vectors of adjacent surface parts.

Parameters **adjacentObjects** (*list(CityGMLObject.SurfacePartsContainer)*) – List of adjacent building components.

Returns **projectionVectors** – List of projection vectors to project the length of geometries on the vector and calculate the projected length.

Return type list(numpy.ndarray)

GeometricAnalysis.**getVerticalBoundsOfComponent**(*obj*)
Returns the lower and upper bounds of the obj's extent.

Parameters **obj** (*CityGMLObject.SurfacePartsContainer or CityGMLObject.BuildingInstallation*) – Object to extract the vertical bounds of its spatial extent.

Returns

- **z_min** (*float*) – Lower bound of the components extent.
- **z_max** (*float*) – Upper bound of the components extent.

GeometricAnalysis.**heightAdjustmentFactorAdjacentRoofs**(*roof*, *normalVectorInspectedSurfacePart*, *rules=['roofInclination > 70.0: 1.0', 'roofInclination > 45.0 or lengthRatio > 0.5: 0.333', 'default: 0.0']*)
Returns the adjustment factor depending on the roof configuration.

Parameters

- **roof** (*CityGMLObject.RoofSurface*) – The roof comes from the component's surface part edge that is currently inspected.
- **normalVectorInspectedSurfacePart** (*numpy.ndarray*) – The plane normal vector of the inspected surface part (not necessarily normalized).
- **rules** (*list(str), optional*) – Set of rules which are evaluated considering the roof configuration providing the resulting value. The default is ["roofInclination > 70.0: 1.0", "roofInclination > 45.0 or lengthRatio > 0.5: 0.333", "default: 0.0"].

Returns Adjustment factor for the summation of the roof height.

Return type float

GeometricAnalysis.**isDirected2CertainDevelopmentAreaTypes**(*surfacePartLineSegment*, *surfacePartNormal*, *propertyParcel*, *spatialplan*, *targets*)
Returns the target towards which the surface part normal is directed the most. The concerned measure is the measure of line overlap.

Parameters

- **surfacePartLineSegment** (*list(numpy.ndarray) with length = 2*) – Inspected line segment of a surface part that rewuires a clearance space surface.
- **surfacePartNormal** (*numpy.ndarray*) – Normalized plane normal of the surface part.
- **propertyParcel** (*CityGMLObject.PropertyParcel*) – Property parcel of the building.

- **spatialPlan** (CityGMLObject.SpatialPlan) – Spatial plan for the region of the building.
- **targets** (*list(str)*) – List of strings defining the development area types.

Returns **targetTypes** – List of strings which are extracted from the development areas from the spatial plan.

Return type None or list(str)

GeometricAnalysis.**isNotOnPropertyBoundary**(*obj*, *propertyParcel*, *buffer=0.1*)

Returns true if obj has not any components which intersect with the property boundary.

Parameters

- **obj** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Instance of either the one or the other type.
- **propertyParcel** (CityGMLObject.PropertyParcel) – Property parcel of the building.
- **buffer** (*foat, optional*) – Tolerance value in meters. The default is 10**-1.

Returns Returns true if no component of obj intersects with the property boundary, false oterhwise.

Return type bool

GeometricAnalysis.**isRelevantSurfacePart**(*surfacePart*)

Checks if surfacePart is considered relevant for the clearance space computation. Returns false if surfacePart is inside of the convex hull of another surface part of the containing *CityGMLObject.SurfacePartsContainer* and does not have an areal extent inside of the containing other surface part.

Parameters **surfacePart** (CityGMLObject.BuildingSurfacePart) – Surface part that is checked on relevance.

Returns Returns true if the surface part is considered relevant for the clearance space computation.

Return type bool

GeometricAnalysis.**lineOverlap**(*line0*, *line1*)

Returns the overlap in percentage of line1 w.r.t. line0. It must be takedn care of that line0 has extent in the x y plane.

Parameters

- **line0** (*list(numpy.ndarray) with length = 2*) – Line segment described by start and end point.
- **line1** (*list(numpy.ndarray) with length = 2*) – Line segment described by start and end point.

Returns Percentage of line overlap w.r.t. the length of line0.

Return type float

GeometricAnalysis.**lineSegmentTopology**(*lStart*, *lEnd*, *n*, *buildingTICs*, *lineArray*)

For the input line defined by start point lStart and end point lEnd it is computed the positions to retrieve the height from the digital terrain model (DTM). It is analyzed if the line is inside or outside of the polygon enclosed by the terrain intersection curves (TIC). For parts of the line which are located inside of the TIC enclosed area the start and end point of line part are projected on the corresponding TIC which yields the position for the height from the DTM. For parts of line located outside of the TIC polygon the line part is returned as it is.

Parameters

- **lStart** (*numpy.ndarray*) – Start point of the line.

- **lEnd** (*numpy.ndarray*) – End point of the line.
- **n** (*numpy.ndarray*) – Normalized plane normal vector of the surface part which the line defined by `lStart` and `lEnd` belongs to.
- **buildingTICs** (*list(shapely.geometry.linestring.LineString)*) – List of line strings representing the building TICs.
- **lineArray** (*list(list(numpy.ndarray))*) – (Reference to) empty list to be filled with lists containing the positions of line segment parts.

Returns

Return type None.

`GeometricAnalysis.`**`linesOverlap`**(*l0*, *l1*)

Returns true if the lines overlap, false otherwise.

Parameters

- **l0** (*list(numpy.ndarray) with lenght = 2*) – Line segment described by start and end point.
- **l1** (*list(numpy.ndarray) with lenght = 2*) – Line segment described by start and end point.

Returns True if the lines overlap, false otherwise.

Return type bool

`GeometricAnalysis.`**`meanDirection`**(*points*)

Calculates the x y direction of greatest extent of the point collection `points` by evaluating the covariance matrix on eigenvectors.

Parameters **points** (*list(numpy.ndarray)*) – List containing arrays containing the coordinates for each point.

Returns

- **meanPoint** (*list(float)*) – Point described by the averaged x y components of `points`.
- **meanDir** (*numpy.ndarray*) – 2D vector of the direction of greatest variance of the point collection `points`.

`GeometricAnalysis.`**`meanNormalVector`**(*surfaces*)

Returns the normalized mean normal vector of the list of surface parts computed by weighting the normal vectors by the area of the contributing surface parts.

Parameters **surfaces** (*list(CityGMLObject.BuildingSurfacePart)*) – List of instances of :py:class`CityGMLObject.BuildingSurfacePart` or any subclass of :py:class`CityGMLObject.BuildingSurfacePart`.

Returns **meanVector** – Normalized mean normal vector of the set of surface parts.

Return type numpy.ndarray

`GeometricAnalysis.`**`objectPolygonCrossing`**(*polygon*, *obj*)

Analyses if it exists an geometrical topological crossing between `polygon` and `obj`. For this, `polygon` and `obj` are transformed to the base which aligns the 1st and 2nd axis with the plane defined by `polygon`, with a 3rd axis parallel to the face normal of `polygon`.

Parameters

- **polygon** (*shapely.geometry.polygon.Polygon*) – Polygon object.

- **obj** (Any type of shapely geometry allowing direct access to the coords attribute.) – Geometry object.

Returns Returns true if there is a crossing between polygon and obj, false otherwise.

Return type bool

GeometricAnalysis.**objectPolygonIntersection**(*polygon*, *obj*)

Analyses if it exists an geometrical topological intersection between polygon and obj. For this, polygon and obj are transformed to the base which aligns the 1st and 2nd axis with the plane defined by polygon, with a 3rd axis parallel to the face normal of polygon.

Parameters

- **polygon** (*shapely.geometry.polygon.Polygon*) – Polygon object.
- **obj** (Any type of shapely geometry allowing direct access to the coords attribute.) – Geometry object.

Returns Returns true if there is an intersection, false otherwise.

Return type bool

GeometricAnalysis.**openedUpwards**(*hullPolygon*, *differencePolygon*)

Checks if the original polygon which was used to derive hullPolygon and differencePolygon is concave with upward orientation. Returns true if the differencePolygon has a parallel edge to:py:attr:*hullPolygon* which has a zero distance and the cross product of the edge vector and e_z returns a vector with positive y value.

Parameters

- **hullPolygon** (*shapely.geometry.polygon.Polygon*) – Convex hull of the original polygon. Polygon points should be ordered counter-clockwise according to the right-hand-rule.
- **differencePolygon** (*shapely.geometry.polygon.Polygon*) – The geometric difference operation between the convex hull of the original polygon and the original polygon.

Returns Returns true if the concavity is oriented upwards, false otherwise.

Return type bool

GeometricAnalysis.**orderPointsAlongProjectionAxis**(*func*, *refPoint*, *projectionAxis*, *points*)

Orders the points along the projection axis projectionAxis according to the distance from the reference Point refPoint.

Parameters

- **func** (*function*) – Function returning a value used for the sorting.
- **refPoint** (*list(x, y, [, z])*) – List containing the coordinates of the reference point.
- **projectionAxis** (*numpy.ndarray*) – Projection axis vector on which the distances are projected.
- **points** (*list(numpy.ndarray)*) – List of points.

Returns **points** – Ordered point list.

Return type list(numpy.ndarray)

GeometricAnalysis.**pointInList**(*sequence*, *point*, *tol=1e-07*)

Returns true if the difference between point and any entry of sequence is smaller than the tolerance defined by tol.

Parameters

- **sequence** (*list((numpy.ndarray)*) – List containing unique entries.
- **point** (*numpy.ndarray*) – Object to check if a similiar entry already exists in sequence.
- **tol** (*float, optional*) – Tolearance value for the measure of similarity. The default is 10**-7.

Returns Returns true if point is already in sequence, false otherwise.

Return type bool

GeometricAnalysis.**pointList2TICs**(*orderedPointCollection*, *surfacePolygon*)

Turns the the terrain intersection points orderedPointCollection of the surface defined by surfacePolygon into a set of line strings.

Parameters

- **orderedPointCollection** (*list(numpy.ndarray)*) – Ordered list of terrain intersection points.
- **surfacePolygon** (*shapely.geometry.polygon.Polygon*) – Polygon geometry of the thematic surface part intersection the terrain.

Returns **ticLineStrings** – List containing the terrain intersection curves for surfacePolygon.

Return type list(shapely.geometry.linestring.LineString)

GeometricAnalysis.**pointOnLineSegment**(*pA*, *vecA*, *p*, *epsilon=1e-05*)

Checks if point p lays on the line segment defined by pA and vecA.

Parameters

- **pA** (*numpy.ndarray*) – Reference point of the line segment (start point).
- **vecA** (*numpy.ndarray*) – Direction vector of the line segment derived by the vector difference of start and end point of the line (not normalized).
- **p** (*numpy.ndarray*) – Point that is analyzed.
- **epsilon** (*float, optional*) – Error tolerance in meters. The default is 10**-5.

Returns

- **p** (*numpy.ndarray*) – Intersection point, i.e. the input point p.
- **x** (*list(float) with lenght = 2*) – Scaling factors for the line equations in vector format. p is considered to be a line equation with zero vector as direction vector.

GeometricAnalysis.**polygonArea**(*pointList*)

Returns the area of the polygon.

Parameters **pointList** (*list(tuple(x, y, z))*) – Point list of coordinate tuples describing the outer ring of the polygon.

Returns **area** – Area enclosed by the exterior ring described by pointList.

Return type float

GeometricAnalysis.**porchLengthRatioViolation**(*obj*, *rule='(dPorch / dWalls) > (1.0 / 3.0)'*)

Finds the length ratio of obj with respect to the adjacent walls and returns true if the ratio exceeds the threshold rule rule.

Parameters

- **obj** (CityGMLObject.BuildingInstallation) – Building installation of the type 'porch'.
- **rule** (*str, optional*) – Rule defining the length ratio between the porch and the adjacent wall. The default is '(dPorch / dWalls) > (1.0 / 3.0)'.

Returns Returns true if the rule evaluation is positive, false otherwise.

Return type bool

GeometricAnalysis.**porchPropertyBoundaryDistanceViolation**(*obj*, *propertyParcel*, *rule='distance > 2.0'*)

Checks if the porch exceeds the minimum distance to the property boundary.

Parameters

- **obj** (CityGMLObject.BuildingInstallation) – The porch that is analyzed.
- **propertyParcel** (CityGMLObject.PropertyParcel) – The property parcel of the building.
- **rule** (*str, optional*) – Rule defining the violation. The default is 'distance > 2.0'.

Returns The rule is evaluated with the closest distance that is found between the porch and the property parcel.

Return type bool

GeometricAnalysis.**privilegeBuildingInstallation**(*buildingInstallation*, *propertyParcel*)

Finds the line segments on which the building is erected if exising and sets the usage code of the building installation side walls to "privileged". if they meet the constraints.

Parameters

- **buildingInstallation** (CityGMLObject.BuildingInstallation) – Building installation object.
- **propertyParcel** (CityGMLObject.PropertyParcel) – The property parcel of the building.

Returns

Return type None.

GeometricAnalysis.**privilegeBuildingPart**(*buildingPart*, *digitalTerrainModel*, *propertyParcel*, *ruleTargets="target in ['garage']"*, *ruleThreshold='h <= 3.0'*)

Sets all components of buildingPart to "privileged", if the building part class code a target type and exceeding the threshold.

Parameters

- **buildingPart** (CityGMLObject.BuildingPart) – Building part object.
- **digitalTerrainModel** (*tuple(list([x, y]), list(z))*) – The digital terrain model in the near surrounding of the building as coordinate tuples of the grid points with the corresponding height values.
- **propertyParcel** (*CityGML*) – Property parcel of the building.

- **ruleTargets** (*str, optional*) – Rule defining required function code values of nuildingPart. The default is "target in ['garage']".
- **ruleThreshold** (*TYPE, optional*) – Rule defining the constraints that have to be met to privilege the building part. The default is "h <= 3.0".

Returns

Return type None.

GeometricAnalysis.**privilegeComponent**(*component*, *propertyParcel*)

Sets the usage code to "privileged" if the component is located on a property boundary line segment.

Parameters

- **component** (CityGMLObject.SurfacePartsContainer) – Component that is checked if it can be neglected for the purpose of clearance space computation.
- **propertyParcel** (CityGMLObject.PropertyParcel) – The property parcel of the building.

Returns

Return type None.

GeometricAnalysis.**projectPoint2Plane**(*normalVector*, *planePoint*, *point*)

Conducts a projection of the point point on the plane which is defined by normalVector and planePoint. The projection is the intersection point of the line equation in vector format with the plane. The reference point for the line equation is point, the direction vector is the normalized plane normal vector normalVector. The intersection point is the then

$$P_{projected} = P + d \cdot \vec{n}$$

With *d* the signed distance of point to the plane and *n* as normalVector.

Parameters

- **normalVector** (*numpy.ndarray*) – Normal vector of the plane (not necessarily normalized).
- **planePoint** (*tuple(x, y, z)*) – Reference point on the plane.
- **point** (*numpy.ndarray*) – Point to project on the plane.

Returns **projectedPoint** – Projected point on the plane.

Return type numpy.ndarray

GeometricAnalysis.**projectPolygon**(*poly*, *M=None*)

Projects poly on the plane defined by poly. The first to axis lay in the polygon plane, the third axis is parallel to the plane normal vector.

Parameters

- **poly** (*shapely.geometry.polygon.Polygon*) – Polygon object.
- **M** (*numpy.ndarray, optional*) – 3x3 transformation matrix. The default is None.

Returns

- **M** (*numpy.ndarray*) – 3x3 transformation matrix derived by the polygon geometry of poly.
- *shapely.geometry.polygon.Polygon* – Transformed polygon in the new base.

GeometricAnalysis.**propertyBoundaryConstructionLineStrings**(*building*, *propertyParcel*)

Returns the line segments of the property boundary on which there are located building components.

Parameters

- **building** (CityGMLObject.Building) – Building object.
- **propertyParcel** (CityGMLObject.PropertyParcel) – Property parcel of the building.

Returns List of shapely LineStrings.

Return type list(shapely.geometry.linestring.LineString)

GeometricAnalysis.**protrusionViolation**(*obj*, *rule='protrusion > 1.6'*, *targetTypes=<class 'object'>*)

Returns true if obj exceeds the given protrusion threshold w.r.t its adjacent objects. Can be used for porches as well as for structural components like cornices, or roof overhangs, etc.

Parameters

- **obj** (CityGMLObject.SurfacePartsContainer *or* CityGMLObject.BuildingInstallation) – Inspected object.
- **rule** (*str, optional*) – Rule defining the constraints to violate the protrusion threshold. The default is 'protrusion > 1.6'.
- **targetTypes** (*tuple(class0, class1, ..), optional*) – Tuple of types to to search for in the list of adjacent objects of obj. The default is (object).

Returns Returns true if the protrusion rule evaluation yields a positive result (=true).

Return type bool

GeometricAnalysis.**removeDuplicates**(*objectList*, *func*)

Removes duplicate ojects from the list objectList.

Parameters

- **objectList** (*list(any)*) – List containg entries of any type.
- **func** (*function*) – Function to compare two objects returning true if the checks conducted in func yield the result of identical objects.

Returns **objectList** – List without duplicate objects.

Return type list(any)

GeometricAnalysis.**roofConstructionLengthRatio**(*roof*, *normalVector=None*)

Returns the ratio of the accumulated length of the list of adjacent roof constructions w.r.t. to the length of the roof. The projection direction vector on which the coordinates are projected is computed by means of the normal vector of the inspected wall surface.

Parameters

- **roof** (CityGMLObject.RoofSurface) – A roof that is adjacent to a wall surface.
- **normalVector** (*numpy.ndarray, optional*) – The plane normal vector of the inspected surface part (not necessarily normalized). The default is None.

Returns Ratio of the length of the roof constructions w.r.t. the roof.

Return type float

GeometricAnalysis.**roofConstructionViolation**(*obj*, *ruleHeight='hRoofConstruction > hRoof'*, *ruleAreaRatio='(areaRoofConstruction / areaRoof) > 0.5'*, *ruleProtrusion='protrusion'*)

According to 'definition dachgaube.pdf' and 'definition dachaufbauten.pdf' roof construction have to be anal-

ysed if they are indiviudal components or secondary elements which are part of the roof. The criteria are the height, the areal ratio and if there is a protrusion which is exceeding the roof.

Parameters

- **obj** (CityGMLObject.BuildingInstallation) – Building installtion classified as roof construction.
- **ruleHeight** (*str, optional*) – Rule to define a threshold for the height roof constructions w.r.t. to the roof. The default is "hRoofConstruction > hRoof".
- **ruleAreaRatio** (*TYPE, optional*) – Rule to define a threshold for the area of roof constructions w.r.t to the area of the roof. The default is "(areaRoofConstruction / areaRoof) > 0.5".
- **ruleProtrusion** (*TYPE, optional*) – Rule to define a threshold for the protrusion of roof constructions from the exterior walls w.r.t the roof. The default is "protrusion".

Returns Returns true if any rules are violated.

Return type bool

GeometricAnalysis.**setAll2Relevant**(*obj*)

Sets all obj components usage code to "relevantForClearanceSpace" except surfaces which have explicetly the usage code "notRelevantForClearanceSpace".

Parameters **obj** (CityGMLObject.SurfacePartsContainer) – Instance of the class :py:class`CityGMLObject.SurfacePartsContainer` or of any subclass of :py:class`CityGMLObject.SurfacePartsContainer` .

Returns

Return type None.

GeometricAnalysis.**terrainIntersectionCurve**(*componentList*, *dtmPoints*)

Copmutes the terrain intersection curves (TIC) for each of the components in componentList.

Parameters

- **componentList** (*list(*CityGMLObject.SurfacePartsContainer*)*) – Buidling components for which the terrain intersection is computed.
- **dtmPoints** (*list(shapely.geometry.point.Point)*) – Points defining the digital terrain model (DTM).

Returns **buildingTIC** – List containing the TICs of the building.

Return type list(shapely.geometry.linestring.LineString)

GeometricAnalysis.**ticIntersection**(*line*, *M*, *surfacePartPolygon_newBase*, *ticList*)

Copmutes the intersection of line with any of the terrain intersection curves (TIC) of the building.

Parameters

- **line** (*list(tuple(x, y, z)) with lenght = 2*) – List of two coordinate tuples defining a line.
- **M** (*numpy.ndarray*) – 3x3 transformation matrix to transform the TICs into the new base defined by the plane of the surface part from which the line segment line comes.
- **surfacePartPolygon_newBase** (*shapely.geometry.polygon.Polygon*) – Transformed surface part polygon which line is part of.
- **ticList** (*list(shapely.geometry.linestring.LineString)*) – List of the building's terrain intersection curves.

Returns **linePartitions** – Line partitions cut by the TICs returned as list of lists containing the start and end point of each line partition.

Return type list(list(numpy.ndarray))

GeometricAnalysis.**tinInsidePolygon**(*points*, *clipper*)

GeometricAnalysis.**transformObjectGeometries**(*obj*, *T*)
Returns a list of transformed polygons of an object.

Parameters

- **obj** (CityGMLObject.SurfacePartsContainer) – Instance of :py:class`CityGMLObject.SurfacePartsContainer` or of any subclass of :py:class`CityGMLObject.SurfacePartsContainer`.
- **T** (*numpy.ndarray*) – 3x3 transformation matrix.

Returns List of transformed polygons.

Return type list(shapely.geometry.polygon.Polygon)

GeometricAnalysis.**transformPoints**(*mappingMatrix*, *points*)
Returns a list of transformed points. 'points' is a list of coordinate tuples of the form [(x, y[, z]), . . .].

Parameters

- **mappingMatrix** (*numpy.ndarray*) – Transformation mattrix to perfom the transformation of the points in `points`.
- **points** (*list(tuple(x, y, z))*) – List of coordinate tuples.

Returns **newCoordRep** – List of arrays containing the points in the new coordinate representation.

Return type list(np.ndarray)

GeometricAnalysis.**transformationMatrix**(*baseVectorsS0*, *baseVectorsS1*)
Returns a transformation matrix to transform points from System S0 to System S1.

Parameters

- **baseVectorsS0** (*list(numpy.ndarray)*) – Lists of base vectors ordered according to their axis indices, starting from i=0.
- **baseVectorsS1** (*list(numpy.ndarray)*) – Lists of base vectors ordered according to their axxis indices, starting from i=0.

Returns Transformation matrix for the transformation from System S0 to System S1.

Return type np.ndarray

GeometricAnalysis.**uniteSurfaces**(*listOfSurfacePartsContainers*, *bufferValue=1.0*)
Computes a union as well as a buffered union of the provided :py:class`CityGMLObject.SurfacePartsContainer` objects in `listOfSurfacePartsContainer`.

Parameters

- **listOfSurfacePartsContainers** (*list(*CityGMLObject.SurfacePartsContainer*)*) – List of :py:class`CityGMLObject.SurfacePartsContainer` objects.
- **bufferValue** (*float, optional*) – Buffer value for the amount of buffering of `polygonUnion`. The default is 1.0.

Returns

- **polygonUnion** (*shapely.geometry.polygon.Polygon*) – Union of all the geometries contained in the `listOfSurfacePartsContainer`.
- **polygonUnionBuffer** (*shapely.geometry.polygon.Polygon*) – Buffered union of all the geometries contained in the `listOfSurfacePartsContainer`.

`GeometricAnalysis.`**`virtualTIC`**(*lineSegment*, *n*, *buildingTICs*, *gridPoints*, *gridValues*)

Returns the height of the start and end point of the line segment `lineSegment`. If the line is within the area enclosed by the terrain intersection curves (TIC) `buildingTICs` the height value at the TIC s used. If the line is located outside of the enclosed area of the buidling TIC the height value is interpolated from the digital terrain model (DTM).

Parameters

- **`lineSegment`** (*`list([x, y, z]) with length = 2`*) – List containing list holding the points of the line.
- **`n`** (*`numpy.ndarray`*) – Not normalized plane normal vector of the polygon to which `lineSegment` belongs.
- **`buildingTICs`** (*`list(shapely.geometry.linestring.LineString)`*) – List of line strings representing the building TICs.
- **`gridPoints`** (*`list([x, y])`*) – List of 2D positions of the DTM.
- **`gridValues`** (*`list(float)`*) – List containing the height values for the corresponding 2D positions in `gridPoints`.

Returns Line representing `lineSegment` projected on the DTM.

Return type shapely.geometry.linestring.LineString

CHAPTER

SIX

RULEENGINE MODULE

class RuleEngine.**RuleEngine**(*operations*, *decisionTreeDOMObject*)

Bases: object

Rule engine class to which walks through a decision tree based on the result of a set of function which are conducted at each node.

operations

A dicitionary containing the operations which are called at each node. The format is e.g.:

```
functionPool_RuleEngine = {
    "func0": lambda obj: isinstance(obj, citygmlobj.BuildingPart),
    "func1": function_from_the_same_module,
    "operation0": externalModule.function_from_external_module
   };
```

Type dict(str : function)

decisionTreeDOMObject

DOM object of the decision tree.

Type xml.dom.minidom.Document

objectOfInterest

Object on which the operations at the respective decision tree node are conducted.

Type object

__getNodes(*childNodes*)

Extracts the necessary nodes for the further walk-through:

- <operation/>: Node holding the name of the operation which will be conducted on the object of interest
- <parameters/>: Node holding the parameters of the operation which will be conducted on the object of interest
- <true/>: Node holding the path for a positive examination result (=true) of the operation
- <false/>: Node holding the path for a negative examination result (=false) of the operation

Parameters **childNodes** (*list(xml.dom.minidom.Element)*) – List of nodes contained by the parent node.

Returns

- **trueNode** (*xml.dom.minidom.Element*) – Node holding the path for a positive examination result (=true) of the operation.

- **falseNode** (*TYPE*) – Node holding the path for a negative examination result (=false) of the operation.
- **operationNode** (*xml.dom.minidom.Element*) – Node holding the name of the operation which will be conducted on the object of interest.
- **parametersNode** (*xml.dom.minidom.Element*) – Node holding the parameters of the operation which will be conducted on the object of interest.

__walkThroughTree(*node*)

Recursive function traversing throught the decision tree and taking the path depending on the operation result at each node.

Parameters **node** (*xml.dom.minidom.Element*) – Current node of the decision tree.

Returns Result of the decision tree at leaf level.

Return type bool

setObjectOfInterest(*obj*)

Sets the object of interest.

Parameters **obj** (*object*) – Can be any object which the functions in `operationDictionary` are capable of to deal with.

Returns

Return type None.

startRuleEngine()

Starts the rule engine for the walk throught the decision tree.

Returns Returns the result of the decision tree.

Return type bool

CHAPTER

SEVEN

WORLDFILEREADER MODULE

class WorldFileReader.**WorldFileReader**(*filename*)

Bases: object

Reader to extract the parameters from a world file.

filename

Path to the world file.

Type str

getParameters()

Extracts the world file parameters.

Returns List of the parameters in the world file.

Return type list(float)

CHAPTER

EIGHT

SAMPLECODE.PY

```
# -*- coding: utf-8 -*-

import xml.etree.ElementTree as ET;
import xml.dom.minidom as XML;

import matplotlib.pyplot as plt;
from matplotlib import animation;
from mpl_toolkits.mplot3d import Axes3D;
from mpl_toolkits.mplot3d.art3d import Poly3DCollection;

import numpy as np;
from shapely.geometry import Polygon, Point, LinearRing, box, MultiPoint,␣
↪MultiPolygon;
from shapely.ops import unary_union;
from shapely.geometry.polygon import orient;
from shapely import affinity;
import alphashape;
import re;
from functools import partial;

import CityGMLObject as citygmlobj;
import CityGMLProcessing as citygml;
import DTM as dtm;
import GeometricAnalysis as ga;
import ClearanceSpaceInspector as csi;
import RuleEngine as rEn;
import WorldFileReader as wldReader;


# Functions
def visualizeFaces(surfaceList, axis):
    for surface in surfaceList:
        renderFaces(surface.surfaceParts, axis);

def renderFaces(faceList, axis):

    vertices = [];
    for face in faceList:
        # Polygon
        list_temp = list(face.geometry.exterior.coords);
        vertices.append(list_temp);
        collection = Poly3DCollection(vertices);
        collection.set_color(face.appearance);
        collection.set_edgecolor('k');
```

(continues on next page)

(continued from previous page)

```
        axis.add_collection3d(collection);
        vertices.clear();

        if isinstance(face, citygmlobj.BuildingSurfacePart):
            renderFaces(face.clearanceSpaceSurfaces, axis);

def points2Polygon(lineStringList):
    points = [];

    for lineString in lineStringList:
        points = points + [p[0:2] for p in list(lineString.coords)];

    shape = alphashape.alphashape(points, 0.1);

    return shape;

def centroid(polygon):
    x_mean = np.mean(polygon.exterior.xy[0]);
    y_mean = np.mean(polygon.exterior.xy[1]);
    z_mean = np.mean([c[2] for c in list(polygon.exterior.coords)]);

    return np.array([x_mean, y_mean, z_mean]);

# ============================================================================
#                   SCRIPT STARTS HERE
# ============================================================================
worldfilename = "./transformation.wld";

functionPool_Preprocessing = {
                "getTransformationParameters": wldReader.WorldFileReader.
↪getParameters,
                "initialiseRuleEngine": None,
                "readSpatialPlanParameters": None,
                "setBuildingNamespaceAndCityGMLVersion": citygml.
↪setNamespacesAndCityGMLVersion,
                "extractBuilding": citygml.extractElement,
                "createTopologies": ga.createTopologies,
                "aggregateBuildingInstallations": ga.aggregateBuildingInstallations,
                "createDTMBuffer": ga.createDTMBuffer,
                "terrainIntersectionCurve": ga.terrainIntersectionCurve
                };

citygml.transformationParams[2] = 714268.3; # .3
citygml.transformationParams[5] = 5322522.7; # .7
citygml.transformationParams[6] = 547.85;

# Read world file
parameters = functionPool_Preprocessing["getTransformationParameters"](wldReader.
↪WorldFileReader(worldfilename));

citygml.transformationParams = parameters;

domObject = XML.parse('./DachgaubeClosure-V1_IDs-localCodeLists.gml');1

# PROGRAM WORKS ONLY WITH INLINE PROFILE -> SEE CITYGML 2.0 SPECIFICATION
functionPool_Preprocessing["setBuildingNamespaceAndCityGMLVersion"](domObject)
```

(continues on next page)

(continued from previous page)

```
# Extract the building from the DOM
building = functionPool_Preprocessing["extractBuilding"](domObject);

# Unlink the DOM object since the extraction has finished
domObject.unlink();

# Analyse the topologies of the building components -> component graph
functionPool_Preprocessing["createTopologies"](building);

# Aggregate adjacent building installations which are of different priority
functionPool_Preprocessing["aggregateBuildingInstallations"](building.
↪buildingInstallations);

# Each line segment is projected on the x-y-plane
building_XYProjected, buildingDTMBuffer = functionPool_Preprocessing["createDTMBuffer
↪"](building.unpackedComponentsAsList(), bufVal=4.0);

# Terrain intersection curve
dtmBuildingPoints = dtm.readDTM("Trackline.xyz", buildingDTMBuffer);
dtmBuildingDataPoints, dtmBuildingDataPointValues = dtm.
↪dtmPointsAsGridPoints(dtmBuildingPoints);
buildingTIC = functionPool_Preprocessing["terrainIntersectionCurve"](building.
↪unpackedComponentsAsList(), dtmBuildingPoints); # buildingComponents␣
↪[wallSurfaces[2]] [wallSurfaces[0]]
# buildingTIC = [];

# DTM
digitalTerrainModel = (dtmBuildingDataPoints, dtmBuildingDataPointValues);

# Property boundary
buildingPropertyParcel = citygmlobj.PropertyParcel("", buildingDTMBuffer);
# buildingPropertyParcel = citygmlobj.PropertyParcel("", building_XYProjected);
# buildingPropertyParcel = citygmlobj.PropertyParcel("", building.groundSurfaces[0].
↪surfaceParts[0].geometry);

plt.figure();
plt.fill(*buildingPropertyParcel.geometry.exterior.xy);
plt.plot(*buildingPropertyParcel.geometry.exterior.xy, 'k-.');
plt.plot(*buildingPropertyParcel.geometry.exterior.xy, 'ko');
plt.plot(*building.groundSurfaces[0].surfaceParts[0].geometry.exterior.xy, 'r');
plt.show();

# Spatial plan and public spaces
# Die Tiefe der Abstandsflächen beträgt 0,4 H, mindestens 3. In Gewerbe- und␣
↪Industriegebieten genügt eine Tiefe von 0,2 H....
# Means that in any other development areas the factor is 0.4 besides the named␣
↪exceptions
spatialPlanFactors = {
    "öffentlicheVerkehrsfläche": 0.2,
    "öffentlicheWasserfläche": 0.2,
    "öffentlicheGrünfläche": 0.2,
    "Kleinsiedlungsgebiet": 0.4,
    "reinesWohngebiet": 0.4,
    "allgemeinesWohngebiet": 0.4,
    "besonderesWohngebiet": 0.4,
    "Dorfgebiet": 0.4,
    "Mischgebiet": 0.4,
```

(continues on next page)

(continued from previous page)

```
    "urbanesGebiet": 0.4,
    "Kerngebiet": 0.4,
    "Gewerbegebiet": 0.2,
    "Industriegebiet": 0.2,
    "Sondergebiet": 0.4
    };

devArea0 = citygmlobj.SpatialPlanDevelopmentArea("", buildingDTMBuffer, "Kerngebiet");
 ↪ # Industriegebiet buildingDTMBuffer.buffer(20.0, cap_style=3, join_style=2)
yTranslation = buildingPropertyParcel.geometry.bounds[-1] - buildingPropertyParcel.
↪geometry.bounds[1];
pubSpace_poly = affinity.affine_transform(buildingPropertyParcel.geometry, [1.0, 0.0,␣
↪0.0, 1.0, 0.0, yTranslation]);
# pubSpace0 = citygmlobj.SpatialPlanDevelopmentArea("", pubSpace0_poly,
↪"öffentlicheWasserfläche");
pubSpace0 = citygmlobj.SpatialPlanDevelopmentArea("", orient(affinity.scale(pubSpace_
↪poly, xfact=0.5, origin=pubSpace_poly.bounds[0:2])), "öffentlicheWasserfläche");
pubSpace1 = citygmlobj.SpatialPlanDevelopmentArea("", orient(affinity.scale(pubSpace_
↪poly, xfact=0.5, origin=(pubSpace_poly.bounds[2], pubSpace_poly.bounds[1]))),
↪"Mischgebiet");
spatialPlan = citygmlobj.SpatialPlan([devArea0, pubSpace0, pubSpace1],␣
↪spatialPlanFactors);

plt.figure();
plt.fill(*devArea0.geometry.exterior.xy);
plt.fill(*pubSpace0.geometry.exterior.xy, 'g');
plt.fill(*pubSpace1.geometry.exterior.xy, 'y');
plt.plot(*buildingPropertyParcel.geometry.exterior.xy, 'k-.');
plt.plot(*buildingPropertyParcel.geometry.exterior.xy, 'ko');
plt.plot(*building.groundSurfaces[0].surfaceParts[0].geometry.exterior.xy, 'r');
plt.show();

# Initialize CSI
clearanceSpaceInspector = csi.ClearanceSpaceInspector(spatialPlan,␣
↪buildingPropertyParcel, building, digitalTerrainModel, buildingTIC);

# Set parameters
clearanceSpaceInspector.setClearanceSpaceConfiguration();
clearanceSpaceInspector.setDefaultAdjustmentFactor();
clearanceSpaceInspector.setMinimumClearanceSpaceDepth();

# Set consequence function
clearanceSpaceInspector.setConsequenceFunction(csi.ClearanceSpaceInspector.
↪calculateClearanceSpace);

# Check if components can be privileged
clearanceSpaceInspector.privilegeComponents();

# Compose the function set for the rule engine
functionPool_RuleEngine = {
                "isBuildingPart": lambda obj: isinstance(obj, citygmlobj.
↪BuildingPart),
                "isBuildingInstallation": lambda obj: isinstance(obj, citygmlobj.
↪BuildingInstallation),
                "isStructuralElement": lambda obj: obj.functionCode ==
↪"structuralElement",
                "isRoofConstruction": lambda obj: obj.functionCode ==
↪"roofConstruction",
```

(continues on next page)

(continued from previous page)

```
                "isPorch": lambda obj: obj.functionCode == "porch",
                "isNotPrivilegedComponent": lambda obj: obj.usageCode ==
↪"relevantForClearanceSpace",
                "isNotOnPropertyBoundary": ga.isNotOnPropertyBoundary,
                "exceedsStructuralElementThresholds": partial(ga.
↪genericExceedsThresholds, checkingFunctions=[partial(ga.protrusionViolation, rule=
↪'protrusion > 1.5')], positiveConsequenceFunc = ga.setAll2Relevant),
                "exceedsRoofConstructionThresholds": partial(ga.
↪genericExceedsThresholds, checkingFunctions=[ga.roofConstructionViolation]),
                "exceedsPorchThresholds": partial(ga.genericExceedsThresholds,␣
↪checkingFunctions=[ga.porchLengthRatioViolation, ga.protrusionViolation,
                                                                partial(ga.
↪porchPropertyBoundaryDistanceViolation, propertyParcel=buildingPropertyParcel)]),
                "porchLengthRatioViolation": ga.porchLengthRatioViolation,
                "exceedsProtrusionThreshold": ga.protrusionViolation,
                "roofConstructionViolation": ga.roofConstructionViolation,
                "roofConstructionLengthRatio": ga.roofConstructionLengthRatio,
                "clearanceSpaceInspection": partial(csi.ClearanceSpaceInspector.
↪recursiveInspection, self=clearanceSpaceInspector)
               };

# Read the descision tree
decisionTree = XML.parse('./decisionTree.xml');

# Initialize rule engine
ruleEngine = rEn.RuleEngine(functionPool_RuleEngine, decisionTree);

# Set the rule engine in the CSI
clearanceSpaceInspector.setRuleEngine(ruleEngine);

# Start clearance space inspection
clearanceSpaceInspector.inspectBuilding4ClearanceSpaces();

building.buildingClearanceSpace();
totalClearanceSpace = building.clearanceSpace.geometry;

plt.figure();
if isinstance(totalClearanceSpace, MultiPolygon):
    for g in totalClearanceSpace.geoms:
        plt.plot(*g.exterior.xy, 'k');
        plt.fill(*g.exterior.xy);
else:
    plt.plot(*totalClearanceSpace.exterior.xy, 'k');
    plt.fill(*totalClearanceSpace.exterior.xy);
plt.show();

# For visualization purposes
buildingSurroundingsBuffer = box(*buildingDTMBuffer.bounds).buffer(1.0, cap_style=3,␣
↪join_style=2);
dtmSurrounding = dtm.readDTM("Trackline.xyz", buildingSurroundingsBuffer);

buildingTin = dtm.createTIN(dtmBuildingPoints, clippingObject=None);

ticPoly = points2Polygon(buildingTIC);

#%%
fig = plt.figure();
```

(continues on next page)

(continued from previous page)

```
plt.grid();
ax = plt.axes(projection='3d');
for c in building.unpackedComponentsAsList():
    for s in c.surfaceParts:
        ax.plot3D(s.geometry.exterior.xy[0], s.geometry.exterior.xy[1], [p[2] for p␣
↪in list(s.geometry.exterior.coords)], "r");

        for interior in s.geometry.interiors:
            ax.plot3D(interior.xy[0], interior.xy[1], [p[2] for p in list(interior.
↪coords)], "b-");

        pointList = list(s.geometry.exterior.coords);
        n = ga.findPlaneNormal(pointList);

        p_centroid = centroid(s.geometry);

        ax.quiver(*p_centroid, *n, normalize=True);

        for i in range(0, len(pointList) - 1, 1):
            line = [pointList[i], pointList[i + 1]];
            virtualTIC = None;
            # virtualTIC = citygml.virtualTIC(line, n, ticPolygon,␣
↪dtmBuildingDataPoints, dtmBuildingDataPointValues);
            if virtualTIC is not None:
                ax.plot3D(virtualTIC.xy[0], virtualTIC.xy[1], [p[2] for p in␣
↪list(virtualTIC.coords)]);

for i in range(0, len(buildingTIC), 1): #  range(2, 3)
    ax.plot3D(buildingTIC[i].xy[0], buildingTIC[i].xy[1], [z[2] for z in␣
↪list(buildingTIC[i].coords)]);
    ax.scatter(buildingTIC[i].xy[0], buildingTIC[i].xy[1], [z[2] for z in␣
↪list(buildingTIC[i].coords)]);

plt.show();


tin = dtm.createTIN(dtmSurrounding, ticPoly);

# ================================================================================
#                                 RENDERING
# ================================================================================
fig0 = plt.figure();
# ax0 = fig0.add_subplot(111, projection='3d');
ax0 = fig0.gca(projection='3d');
# ax0 = Axes3D(fig0);

ax0.set_xlim(buildingDTMBuffer.bounds[0], buildingDTMBuffer.bounds[2]);
ax0.set_ylim(buildingDTMBuffer.bounds[1], buildingDTMBuffer.bounds[3]);
ax0.set_zlim(np.min(dtmBuildingDataPointValues) - 10, np.
↪max(dtmBuildingDataPointValues) + 10);

def init():
    visualizeFaces(building.unpackedComponentsAsList(), ax0);
    dtm.render(ax0, tin);

    plt.show();
    return fig0;
```

(continues on next page)

(continued from previous page)

```
init();

def animate(i):
    # azimuth angle : 0 deg to 360 deg
    ax0.view_init(elev=-5, azim=i*4);
    return fig0;

# for angle in range(0, 360):
#     ax0.view_init(-5, angle);
#     plt.draw();
#     plt.pause(.001);

# plt.show();

# ani = animation.FuncAnimation(fig0, animate, init_func=init, frames=90,␣
↪interval=50);

# fn = 'rotate_azimuth_angle_3d_surf';
# ani.save(fn+'.gif',writer='imagemagick', fps=500/50);
```

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

c

CityGMLObject, 1
CityGMLProcessing, 9
ClearanceSpaceInspector, 15

d

DTM, 19

g

GeometricAnalysis, 21

r

RuleEngine, 39

w

WorldFileReader, 41

INDEX

Symbols

__getNodes() (*RuleEngine.RuleEngine method*), 39
__setClearanceSpaceDepth() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 15
__setSpatialPlanAdjustmentFactor() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 15
__walkThroughTree() (*RuleEngine.RuleEngine method*), 40

A

adjacentObjects (*CityGMLObject.CityGMLObject attribute*), 3
adjustmentFactorRoof (*CityGMLObject.ClearanceSurface attribute*), 4
adjustmentFactorSpatialPlan (*CityGMLObject.ClearanceSurface attribute*), 4
aggregateBuildingInstallations() (*in module GeometricAnalysis*), 21
aggregatedBuildingInstallations (*CityGMLObject.BuildingInstallation attribute*), 2
aggregateSurfaces() (*in module GeometricAnalysis*), 21
areaType (*CityGMLObject.SpatialPlanDevelopmentArea attribute*), 6

B

Building (*class in CityGMLObject*), 1
building (*ClearanceSpaceInspector.ClearanceSpaceInspector attribute*), 15
buildingClearanceSpace() (*CityGMLObject.Building method*), 2
BuildingInstallation (*class in CityGMLObject*), 2
buildingInstallations (*CityGMLObject.Building attribute*), 1
BuildingPart (*class in CityGMLObject*), 3
buildingParts (*CityGMLObject.Building attribute*), 1
BuildingSurfacePart (*class in CityGMLObject*), 3
buildingTICs (*ClearanceSpaceInspector.ClearanceSpaceInspector attribute*), 15

C

calculateClearanceSpace() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 16
checkAdjacency() (*in module GeometricAnalysis*), 21
checkEdgeAdjacency() (*in module GeometricAnalysis*), 21
CityGMLObject
    module, 1
CityGMLObject (*class in CityGMLObject*), 3
CityGMLProcessing
    module, 9
classCode (*CityGMLObject.CityGMLObject attribute*), 3
clearanceDepth (*CityGMLObject.ClearanceSurface attribute*), 4
clearanceSpace (*CityGMLObject.Building attribute*), 2
clearanceSpace() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 16
ClearanceSpaceInspector
    module, 15
ClearanceSpaceInspector (*class in ClearanceSpaceInspector*), 15
clearanceSpaceSurfaces (*CityGMLObject.BuildingSurfacePart attribute*), 3
clearanceSpaceSurfaces (*CityGMLObject.SurfacePartsContainer attribute*), 6
ClearanceSurface (*class in CityGMLObject*), 4
ClosureSurface (*class in CityGMLObject*), 4
ClosureSurfacePart (*class in CityGMLObject*), 4
closureSurfaces (*CityGMLObject.Building attribute*), 1
component2PointCollection() (*in module GeometricAnalysis*), 22
componentSlope() (*in module GeometricAnalysis*),

22
computationDescription (*CityGMLObject.ClearanceSurface attribute*), 4
computeHeightAdjustment() (*in module GeometricAnalysis*), 22
computeLineIntersection3D() (*in module GeometricAnalysis*), 22
createDTMBuffer() (*in module GeometricAnalysis*), 23
createTIN() (*in module DTM*), 19
createTopologies() (*in module GeometricAnalysis*), 23

D

decisionTreeDOMObject (*RuleEngine.RuleEngine attribute*), 39
developmentAreaFactors (*CityGMLObject.SpatialPlan attribute*), 5
developmentAreas (*CityGMLObject.SpatialPlan attribute*), 5
digitalTerrainModel (*ClearanceSpaceInspector.ClearanceSpaceInspector attribute*), 15
dissolveThematicObjects() (*in module CityGMLProcessing*), 9
distanceAlongAxis_Component() (*in module GeometricAnalysis*), 23
distanceAlongAxis_SurfacePart() (*in module GeometricAnalysis*), 23
distanceAlongProjectionAxis() (*in module GeometricAnalysis*), 23
distanceBetweenLines() (*in module GeometricAnalysis*), 24
distancePoint2Plane() (*in module GeometricAnalysis*), 24
Door (*class in CityGMLObject*), 4
doors (*CityGMLObject.ThematicBuildingSurfacePartsContainer attribute*), 7
DoorSurfacePart (*class in CityGMLObject*), 4
DTM
    module, 19
dtmPointsAsGridPoints() (*in module DTM*), 19

E

extractCityObjectRelations() (*in module CityGMLProcessing*), 9
extractCoords() (*in module CityGMLProcessing*), 9
extractElement() (*in module CityGMLProcessing*), 10
extractGeometriesFromComponent() (*in module GeometricAnalysis*), 24
extractGeometry() (*in module CityGMLProcessing*), 10
extractPosFromPosElements() (*in module CityGMLProcessing*), 10
extractPosFromPosListElement() (*in module CityGMLProcessing*), 10
extractSurfaceMembers() (*in module CityGMLProcessing*), 11
extractThematicSurfaces() (*in module CityGMLProcessing*), 11

F

filename (*WorldFileReader.WorldFileReader attribute*), 41
fillingSurfaceTagNamesCityGML3 (*in module CityGMLProcessing*), 11
findAdjacentRoofPartsAbove() (*in module GeometricAnalysis*), 24
findClosestDistance2PropertyBoundary() (*in module GeometricAnalysis*), 25
findElementInChilds() (*in module CityGMLProcessing*), 11
findPlaneNormal() (*in module GeometricAnalysis*), 25
findPropertyBoundarySegments_ONRelation() (*in module GeometricAnalysis*), 25
functionCode (*CityGMLObject.CityGMLObject attribute*), 3

G

genericCheck() (*in module GeometricAnalysis*), 25
genericExceedsThresholds() (*in module GeometricAnalysis*), 26
genericSurfaces (*CityGMLObject.Building attribute*), 1
GenericThematicSurface (*class in CityGMLObject*), 4
GenericThematicSurfacePart (*class in CityGMLObject*), 4
GeometricAnalysis
    module, 21
geometricOperation() (*in module GeometricAnalysis*), 26
geometry (*CityGMLObject.Surface attribute*), 6
getClearanceSpaceDepthAdjustmentFactor() (*in module GeometricAnalysis*), 26
getComponentByIDRelationAndType() (*in module GeometricAnalysis*), 27
getComponents() (*CityGMLObject.Building method*), 2
getComponents() (*CityGMLObject.BuildingInstallation method*), 2
getDescriptionFromDictionary() (*in module CityGMLProcessing*), 12
getDevelopmentAreaTypeOfBuilding() (*in module GeometricAnalysis*), 27

getElementChildNodes() (*in module CityGMLProcessing*), 12
getFirstOfAType() (*in module CityGMLProcessing*), 12
getLengthProjectionVectorsOfAdjacentObjects() (*in module GeometricAnalysis*), 27
getParameters() (*WorldFileReader.WorldFileReader method*), 41
getSurfaceParts() (*CityGMLObject.BuildingInstallation method*), 2
getSurfaceParts() (*CityGMLObject.SurfacePartsContainer method*), 6
getThematicBuildingSurfacePartsContainers() (*in module CityGMLProcessing*), 12
getVerticalBoundsOfComponent() (*in module GeometricAnalysis*), 28
GroundSurface (*class in CityGMLObject*), 4
GroundSurfacePart (*class in CityGMLObject*), 4
groundSurfaces (*CityGMLObject.Building attribute*), 1

H

hasOrientableSurface() (*in module CityGMLProcessing*), 13
heightAdjustmentFactorAdjacentRoofs() (*in module GeometricAnalysis*), 28

I

id (*CityGMLObject.CityGMLObject attribute*), 3
inspectBuilding4ClearanceSpaces() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 16
isDirected2CertainDevelopmentAreaTypes() (*in module GeometricAnalysis*), 28
isNotOnPropertyBoundary() (*in module GeometricAnalysis*), 29
isRelevantSurfacePart() (*in module GeometricAnalysis*), 29

L

lineOverlap() (*in module GeometricAnalysis*), 29
lineSegmentTopology() (*in module GeometricAnalysis*), 29
linesOverlap() (*in module GeometricAnalysis*), 30

M

meanDirection() (*in module GeometricAnalysis*), 30
meanNormalVector() (*in module GeometricAnalysis*), 30
module
    CityGMLObject, 1
    CityGMLProcessing, 9
    ClearanceSpaceInspector, 15
    DTM, 19
    GeometricAnalysis, 21
    RuleEngine, 39
    WorldFileReader, 41

O

objectID (*CityGMLObject.ObjectRelation attribute*), 5
objectOfInterest (*RuleEngine.RuleEngine attribute*), 39
objectPolygonCrossing() (*in module GeometricAnalysis*), 30
objectPolygonIntersection() (*in module GeometricAnalysis*), 31
ObjectRelation (*class in CityGMLObject*), 5
objectRelations (*CityGMLObject.CityGMLObject attribute*), 3
openedUpwards() (*in module GeometricAnalysis*), 31
openingTagNamesCityGML2 (*in module CityGMLProcessing*), 13
operations (*RuleEngine.RuleEngine attribute*), 39
orderPointsAlongProjectionAxis() (*in module GeometricAnalysis*), 31
outerCeilings (*CityGMLObject.Building attribute*), 1
OuterCeilingSurface (*class in CityGMLObject*), 5
OuterCeilingSurfacePart (*class in CityGMLObject*), 5
outerFloors (*CityGMLObject.Building attribute*), 1
OuterFloorSurface (*class in CityGMLObject*), 5
OuterFloorSurfacePart (*class in CityGMLObject*), 5

P

parent (*CityGMLObject.CityGMLObject attribute*), 3
parseCoordinates() (*in module CityGMLProcessing*), 13
pointInList() (*in module GeometricAnalysis*), 31
pointList2TICs() (*in module GeometricAnalysis*), 32
pointOnLineSegment() (*in module GeometricAnalysis*), 32
polygonArea() (*in module GeometricAnalysis*), 32
porchLengthRatioViolation() (*in module GeometricAnalysis*), 32
porchPropertyBoundaryDistanceViolation() (*in module GeometricAnalysis*), 33
privilegeBuildingInstallation() (*in module GeometricAnalysis*), 33
privilegeBuildingPart() (*in module GeometricAnalysis*), 33
privilegeComponent() (*in module GeometricAnalysis*), 34

privilegeComponents() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 16
projectPoint2Plane() (*in module GeometricAnalysis*), 34
projectPolygon() (*in module GeometricAnalysis*), 34
propertyBoundaryConstructionLineStrings() (*in module GeometricAnalysis*), 34
PropertyParcel (*class in CityGMLObject*), 5
propertyParcel (*ClearanceSpaceInspector.ClearanceSpaceInspector attribute*), 15
protrusionViolation() (*in module GeometricAnalysis*), 35

R

readDTM() (*in module DTM*), 19
readDTMAsDataPoints() (*in module DTM*), 19
recursiveInspection() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 16
referenceChildsWithParent() (*CityGMLObject.CityGMLObject method*), 3
relationType (*CityGMLObject.ObjectRelation attribute*), 5
removeChildsFromParent() (*in module CityGMLProcessing*), 13
removeDuplicates() (*in module GeometricAnalysis*), 35
render() (*in module DTM*), 20
resolveCodeListElement() (*in module CityGMLProcessing*), 13
roofConstructionLengthRatio() (*in module GeometricAnalysis*), 35
roofConstructionViolation() (*in module GeometricAnalysis*), 35
roofEdges (*CityGMLObject.BuildingSurfacePart attribute*), 3
roofs (*CityGMLObject.Building attribute*), 1
RoofSurface (*class in CityGMLObject*), 5
RoofSurfacePart (*class in CityGMLObject*), 5
RuleEngine
    module, 39
RuleEngine (*class in RuleEngine*), 39

S

setAll2Relevant() (*in module GeometricAnalysis*), 36
setClearanceSpaceConfiguration() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 17
setConsequenceFunction() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 17
setDefaultAdjustmentFactor() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 17
setMinimumClearanceSpaceDepth() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 17
setNamespacesAndCityGMLVersion() (*in module CityGMLProcessing*), 13
setObjectOfInterest() (*RuleEngine.RuleEngine method*), 40
setRuleEngine() (*ClearanceSpaceInspector.ClearanceSpaceInspector method*), 17
solveXLink() (*in module CityGMLProcessing*), 13
SpatialPlan (*class in CityGMLObject*), 5
spatialPlan (*ClearanceSpaceInspector.ClearanceSpaceInspector attribute*), 15
SpatialPlanDevelopmentArea (*class in CityGMLObject*), 6
startRuleEngine() (*RuleEngine.RuleEngine method*), 40
storeysAboveGround (*CityGMLObject.Building attribute*), 1
Surface (*class in CityGMLObject*), 6
surfaceContainers (*CityGMLObject.BuildingInstallation attribute*), 2
surfaceParts (*CityGMLObject.SurfacePartsContainer attribute*), 6
SurfacePartsContainer (*class in CityGMLObject*), 6

T

terrainIntersectionCurve() (*in module GeometricAnalysis*), 36
ThematicBuildingSurfacePartsContainer (*class in CityGMLObject*), 7
ticIntersection() (*in module GeometricAnalysis*), 36
tinInsidePolygon() (*in module GeometricAnalysis*), 37
transformationMatrix() (*in module GeometricAnalysis*), 37
transformationParams (*in module CityGMLProcessing*), 14
transformObjectGeometries() (*in module GeometricAnalysis*), 37
transformPoints() (*in module GeometricAnalysis*), 37

U

uniteSurfaces() (*in module GeometricAnalysis*), 37
unpackedComponentsAsList() (*CityGMLObject.Building method*), 2
usageCode (*CityGMLObject.CityGMLObject attribute*), 3

V

virtualTIC() (*in module GeometricAnalysis*), 38

W

walls (*CityGMLObject.Building attribute*), 1
WallSurface (*class in CityGMLObject*), 7
WallSurfacePart (*class in CityGMLObject*), 7
Window (*class in CityGMLObject*), 7
windows (*CityGMLObject.ThematicBuildingSurfacePartsContainer attribute*), 7
WindowSurfacePart (*class in CityGMLObject*), 7
WorldFileReader
    module, 41
WorldFileReader (*class in WorldFileReader*), 41

E Anleitung zur Konvertierung eines CAD-Gebäudeentwurfs in ein CityGML-Modell in SketchUp

Anleitung zur Konvertierung eines CAD-Gebäudeentwurfs in ein CityGML-Modell in SketchUp

Beschreibung:

Diese Anleitung dient als Beispiel und Vorlage für die Konvertierung eines zwei-dimensionalen CAD-Gebäudeentwurfs in ein CityGML-Gebäudemodell. Der Arbeitsablauf gliedert sich in zwei Teile. Im ersten Teil wird ein 3D-Modell aus den CAD-Zeichnungen erstellt. Im zweiten Teil wird das 3D-Modell mittels eines SketchUp Plugins in ein CityGML-Gebäudemodell exportiert.

Verwendete Software:

- Trimble SketchUp Pro 2020 (Pro-Version notwendig für dxf/dwg-Import)
- GEORES SketchUp CityGML Plugin (https://github.com/GeoplexGIS/geores, Handbuch: https://docplayer.org/59122411-Produktdokumentation.html)

Beschreibung der Daten:

Die CAD-Zeichnung des Gebäudeentwurfs enthält Ansichten der vier Fassaden des Gebäudes, sowie die Grundrisszeichnungen der Stockwerke.

1 Erstellung eines SketchUp-Modells

1.1 Import des Grundrisses des 0. Stockwerks

Im ersten Schritt wird die CAD-Datei, welche den Grundriss des 0. Stockwerks enthält, in das SketchUp-Projekt importiert. Die CAD-Zeichnung wird von SketchUp als Gruppe importiert. Um unnötige Linien und Objekte zu löschen, muss die Gruppe in ihre Einzelteile aufgelöst werden. Anschließend können störende oder überflüssige Objekte selektiert und entfernt werden. Nachdem der Grundriss in gewünschter Form vorliegt, wird dieser erneut gruppiert. Anschließend wird ein Layer bzw. Tag angelegt, welcher die importierten CAD-Zeichnungen enthalten soll. Dorthin wird der gruppierte Grundriss verschoben.

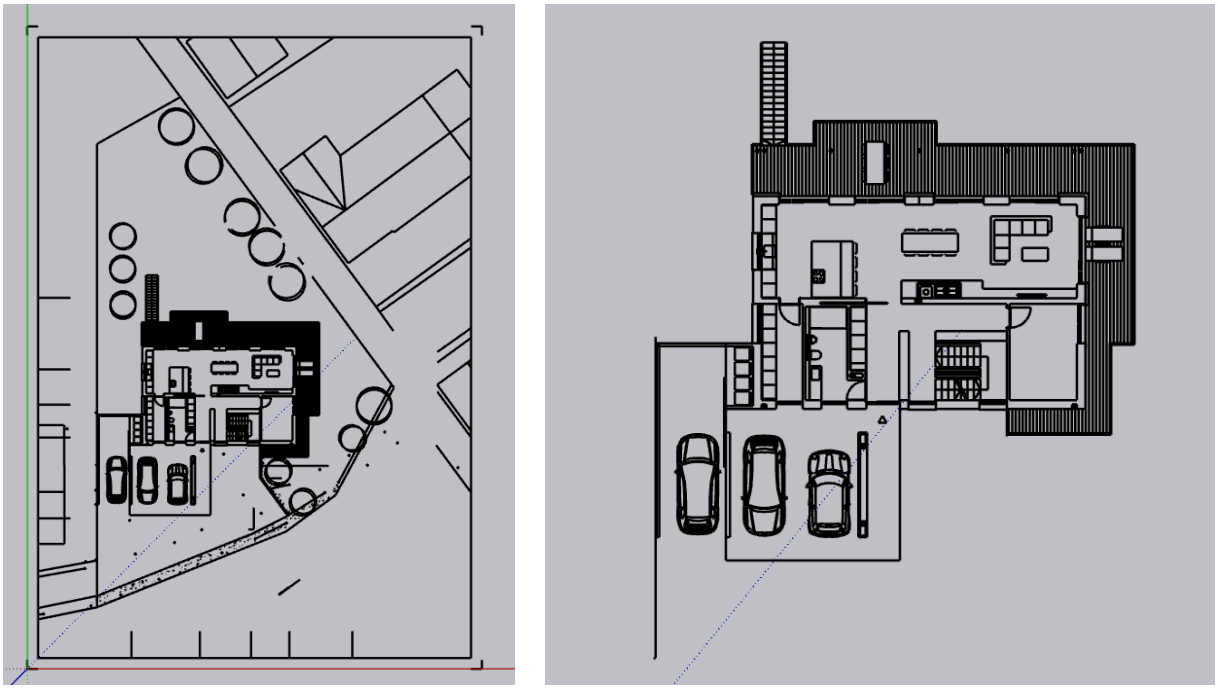


Abbildung 1: Originaler Grundriss (links) und bereinigter Grundriss (rechts).

1.2 Import der Fassaden-Ansichten

1.2.1 Erzeugung der Gebäudehülle

Nun kann die erste Fassaden-Ansicht geladen werden, um die Gebäudehülle zu erzeugen. Welche Ansicht dafür verwendet wird, ist nicht von Relevanz. Auch hier muss die Zeichnung von Störelementen befreit werden. Danach wird die Ansicht gruppiert und in den Layer/Tag der CAD-Zeichnungen verschoben.

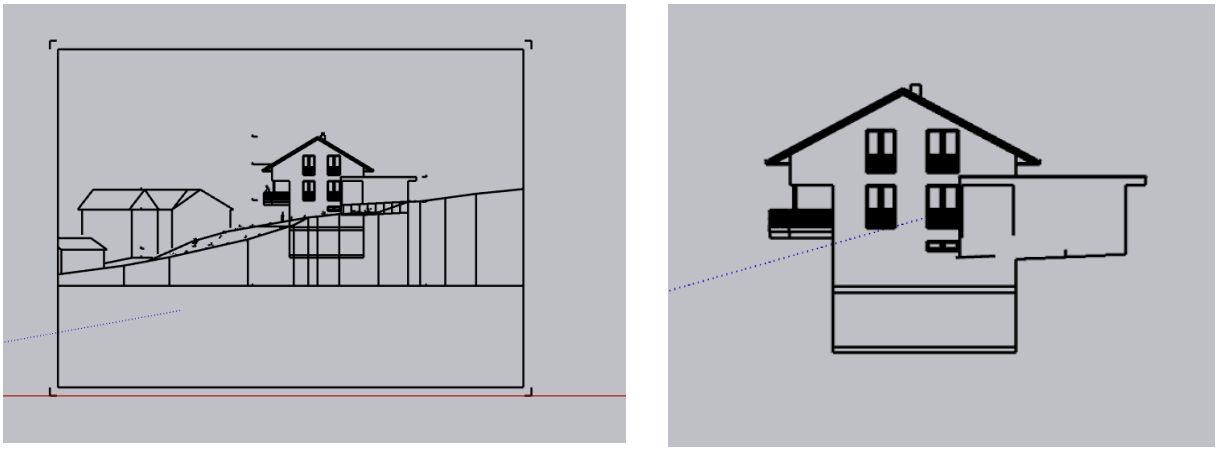


Abbildung 2: Importierte CAD-Zeichnung einer Fassaden-Ansicht (links) und bereinigte CAD-Zeichnung (rehchts).

Anschließend wird die Fassade mittels des Linien-Werkzeugs nachgezeichnet und somit eine Fläche der Fassade erzeugt. Als nächstes werden Fläche und Ansicht über die Rotierfunktion in die Vertikale gebracht und derart verschoben, dass die korrespondierenden Ecken des Grundrisses und der Fassade übereinander zu liegen kommen. Die Einrastfunktion vereinfacht es, Objekte anhand der Eckpunkte anderer Objekte auszurichten.

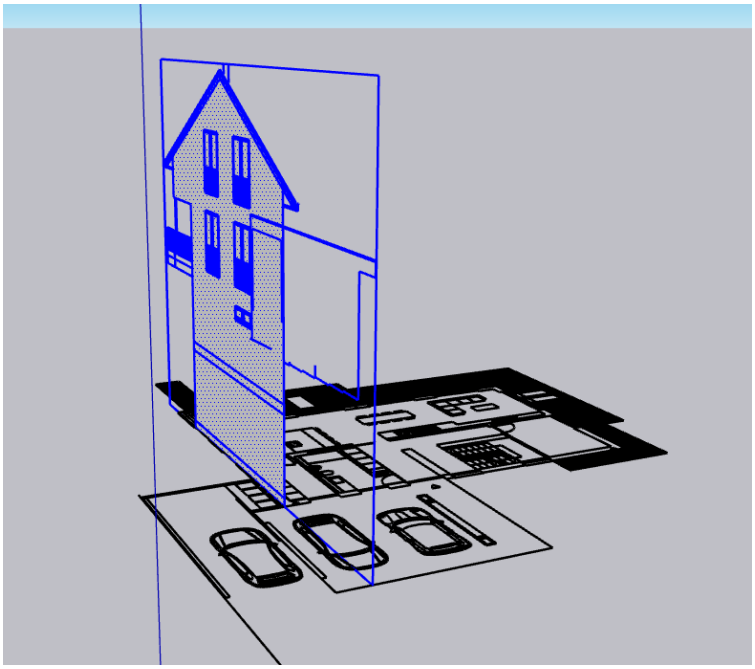


Abbildung 3: Rotierte und verschobene Ansicht der Fassade.

Nun wird über die Drücken/Ziehen-Operation die Fassadenfläche auf die Länge des Gebäudes extrudiert. Auch hier kann die Einrastfunktion helfen, um die Länge genau auf die des Grundrisses abzustimmen. Die dabei entstandenen Gebäudehülle wird anschließend gruppiert und in einen separaten Layer/Tag verschoben.

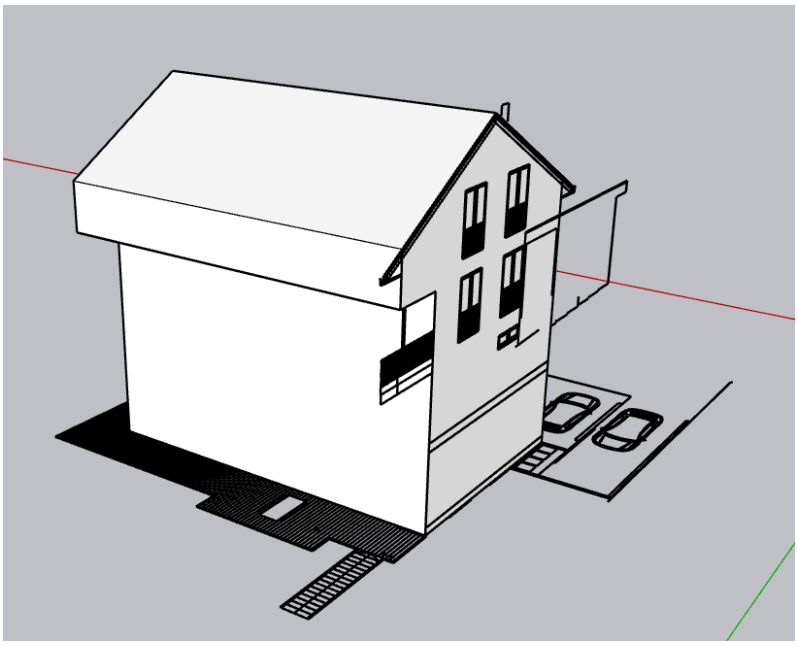


Abbildung 4: Extrudierte Fassaden-Fläche der Referenzansicht.

1.2.2 Anheften der restlichen Fassaden an die Gebäudehülle

Nachdem die Gebäudehülle erzeugt wurde, werden die restlichen Ansichten in das Projekt importiert. Wie vorher werden die Ansichten bereinigt und in den entsprechenden Layer/Tag verschoben. Danach werden sie über Rotieren und Verschieben an die korrekte Position an der Gebäudehülle gebracht. Dabei müssen korrespondierende Punkte angrenzender Ansichten aufeinander zu liegen kommen.

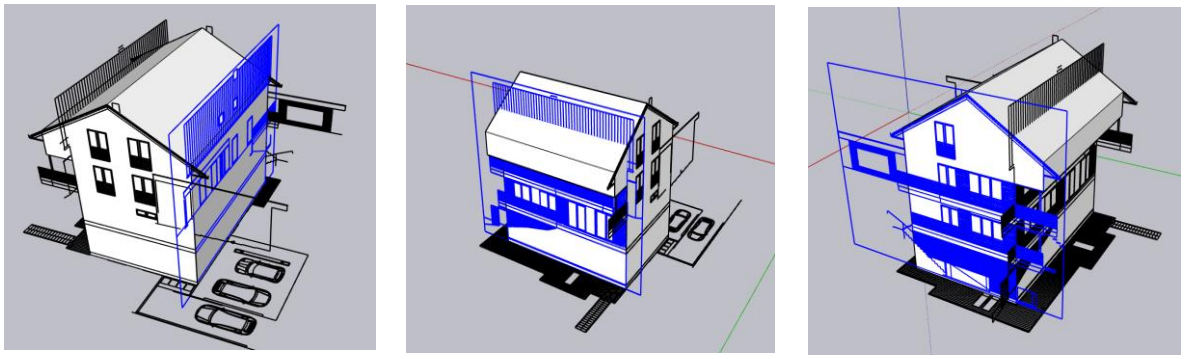


Abbildung 5: Anheften der restlichen Ansichten an die Gebäudehülle.

1.3 Justierung des Grundrisses des 0. Stockwerks

Der Grundriss des 0. Stockwerks ist auf Höhe des Kellerfußbodens. Da dies nicht die korrekte Position ist, wird der Grundriss vertikal verschoben, bis die korrespondierenden Punkte der Referenzansicht des 0.Stocks und des Grundrisses des 0. Stocks übereinander liegen.

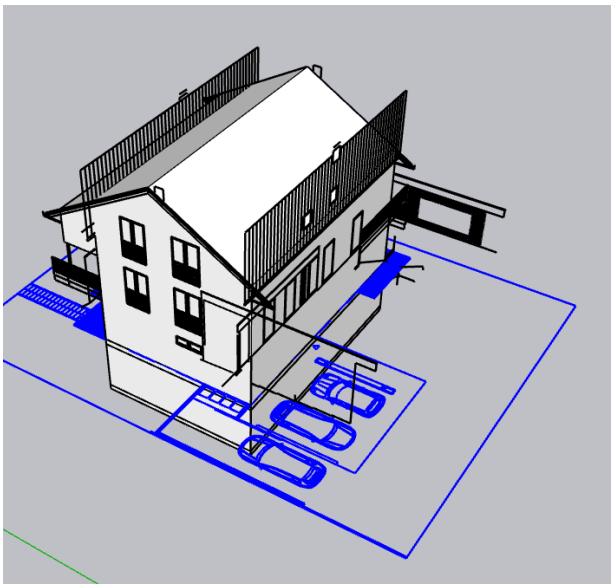


Abbildung 6: Vertikale Justierung der Position des Grundrisses des 0. Stockwerks.

1.4 Import des Grundrisses des 1. Stockwerks

Als nächstes wird die Grundriss-Zeichnung des 1. Stocks importiert. Auch hier findet die Reinigung der CAD-Zeichnung und anschließende Verschiebung in den Layer/Tag für die CAD-Importe statt. Wie auch schon der Grundriss des 0. Stockwerks wird der Grundriss der 1. Etage so verschoben, dass die Etage mit den korrespondierenden Punkten der Referenzfassade des 1. Stockwerks zusammenfällt.

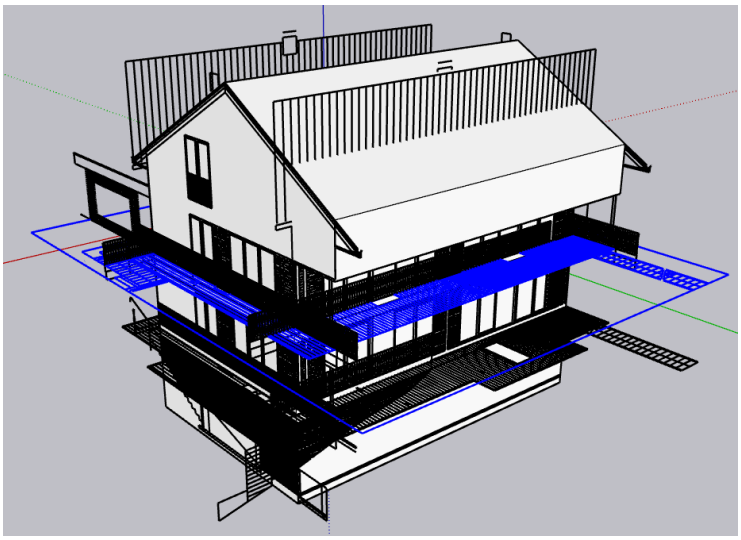


Abbildung 7: Anpassung der Grundriss-Zeichnung des 1.Stockwerks an die Referenzansicht.

1.5 Zusammenführung der Garagenkomponenten

Die Bestandteile der Garage in den CAD-Zeichnungen sind auf die verschiedenen Ansichten verteilt. Um diese zusammenzuführen, werden die jeweiligen Objekte der Garage aus den Ansichten ausgewählt und so verschoben, dass bezüglich der Referenzansicht die korrespondierenden Punkte der Garage übereinander liegen. Abschließend werden die Garagenobjekte gruppiert.

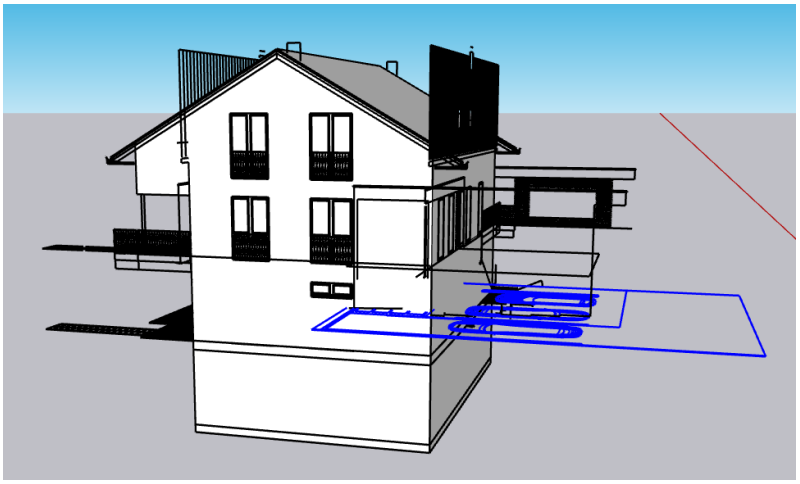


Abbildung 8: Verschieben der Garagenkomponenten aus den verschiedenen Ansichten an die korrekte Position.

1.6 Zeichnen von Türen und Fenster

1.6.1 Türen und Fenster der Fassaden

Die Türen und Fenster der Fassaden können ganz einfach mit dem Linien-Werkzeug nachgezeichnet werden. Da es sich um wiederkehrende Objekte handelt, werden von Türen und Fenstern sogenannte Komponenten erstellt, die dann an die entsprechenden Positionen kopiert werden können. Die entstehenden Flächen auf der Gebäudehülle sind eigenständige Flächen, welche durch Entfernen Löcher in der Gebäudehülle hinterlassen würden.

1.6.2 Dachfenster

Wie in Abbildung 8 zu sehen, ist durch die Parallelprojektion der Ansichten die Dachfläche nur in der Vertikalen verfügbar. Das macht die Rekonstruktion der Dachfenster etwas schwerer, da über Schnitte die ursprüngliche Position zurückverfolgt werden muss.

Zuerst werden wie bei den Fassadenfenstern auch, die Fensterflächen anhand der CAD-Zeichnung nachgezeichnet.

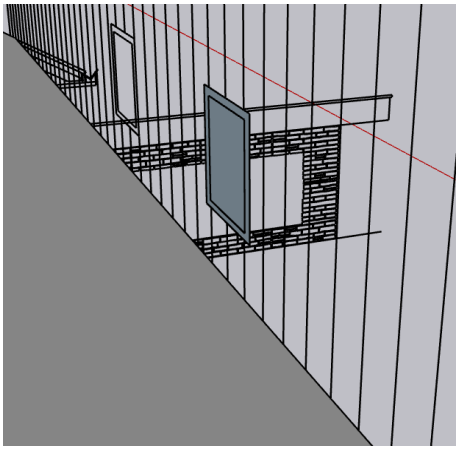


Abbildung 9: Nachzeichnen der Dachfensterflächen.

Als nächstes werden die Fensterflächen extrudiert, und zwar so weit, dass sie in die Gebäudehülle hineinragen.

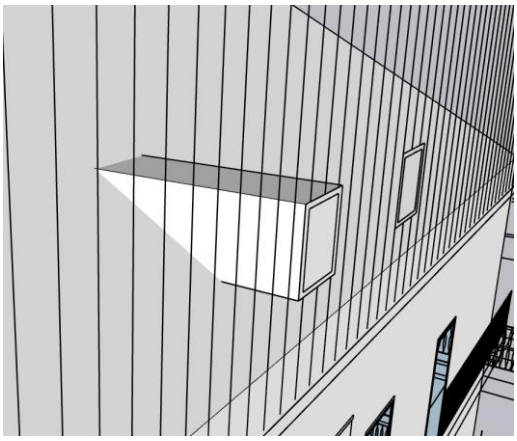


Abbildung 10: Die Flächen der Dachfenster werden so weit extrudiert, dass sie in die Gebäudehülle hineinragen.

Nun können die überflüssigen Linien und Flächen gelöscht werden bis die Schnittfläche auf der Gebäudehülle gezeichnet werden kann.

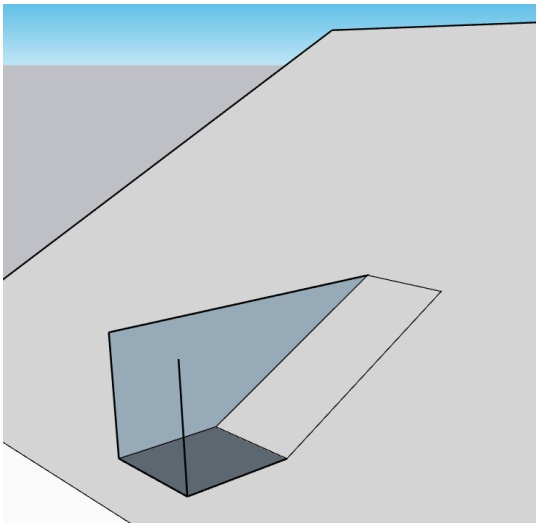


Abbildung 11: Löschen der überflüssigen Komponenten, um die Schnittfläche des Dachfensters zu erhalten.

1.7 Zwischenresultat

In Abbildung 12 sind ist die vorläufige Gebäudehülle mit den eingezeichneten Tür- und Fensterflächen dargestellt. In den nächsten Schritten wird das Gebäude um die noch fehlenden Komponenten (Dachüberstände, Kamin, Balkone, Garage und Treppe) erweitert.

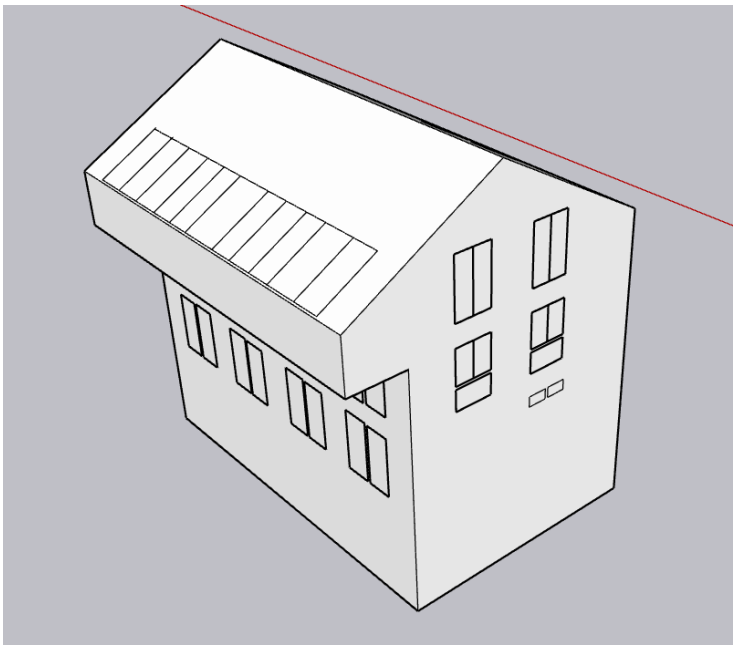


Abbildung 12: Vorläufiges Ergebnis nach der Einzeichnung von Tür- und Fensterflächen in die Gebäudehülle.

1.8 Erstellung des Kamins

Auch die 4 Seiten des Kamins sind auf die vier Ansichten verteilt. Zwei sich nicht gegenüberliegende Kaminseiten werden ausgewählt und mit dem Linien-Werkzeug nachgezeichnet. Danach werden beide Flächen extrudiert, sodass sich die Geometrien verschneiden lassen.

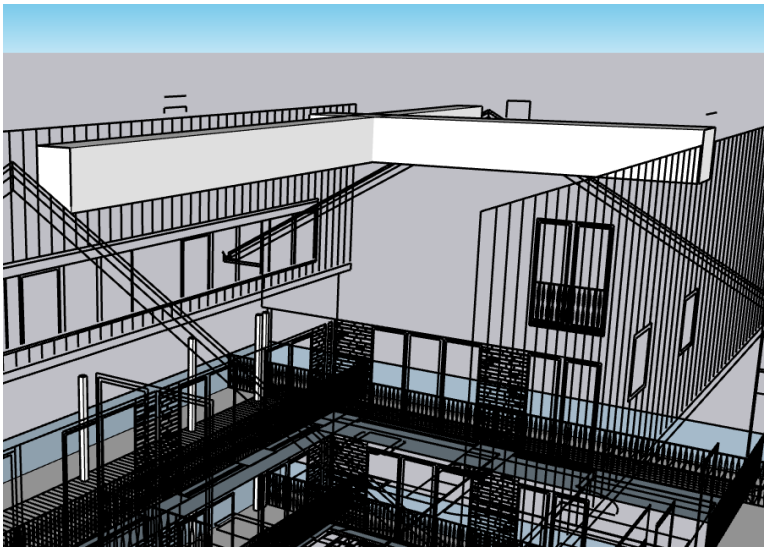


Abbildung 13: Verschneidung der extrudierten Kaminflächen.

Für die Verschneidung der beiden Geometrien müssen beide ausgewählt werden. Nach der Ausführung der Verschneide-Operation, verbleibt die Kamingeometrie. Abschließend werden die Kaminflächen zu einem Objekt gruppiert.

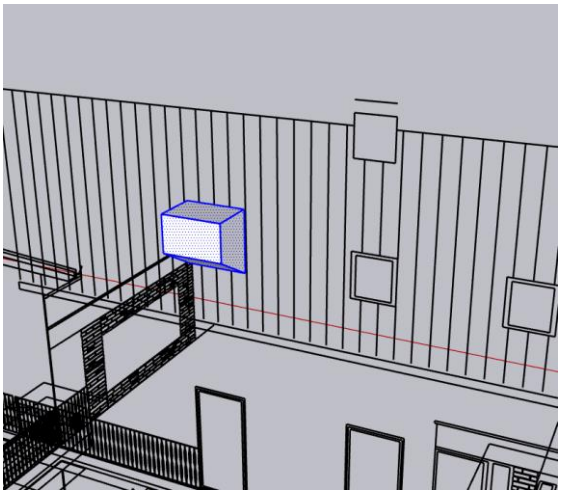


Abbildung 14: Resultierender Kamin aus der Verschneidung der Volumenkörper.

1.9 Erstellung der Garage

Die Erstellung der Hülle der Garage als Gebäudeteil ist analog zum Vorgehen beim Hauptgebäude. Eine Seite der Garage wird als Referenz festgelegt. Diese Seite wird mit dem Linien-Werkzeug nachgezeichnet, sodass eine Fläche entsteht, die auf die Länge der Garage gemäß der CAD-Zeichnungen extrudiert wird. Dasselbe Vorgehen wird auf die Dachfläche der Garage angewandt. Nach Fertigstellung werden alle Garagenflächen zu einem Objekt gruppiert.

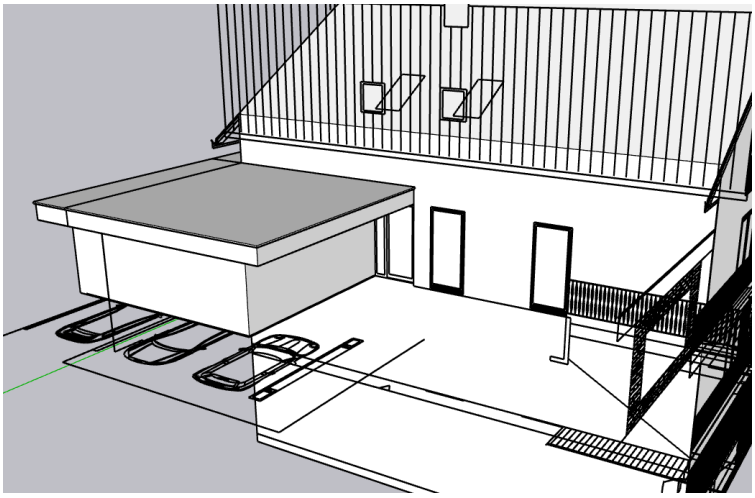


Abbildung 15: Resultierendes Garagenobjekt.

1.10 Erstellung der Dachüberstände

Die Dachüberstände werden mittels zweier angrenzender Gebäudeseiten konstruiert. Die Ansicht der Längsseite liefert das Maß des Überstandes in der Richtung der Längsseite. Die Breitseite liefert die Länge des Dachüberstandes rechtwinklig zur Längsseite des Gebäudes. Je Dachfläche entstehen drei Dachüberstände. Jeder Dachüberstand wird als eigenständiges Objekt gruppiert.

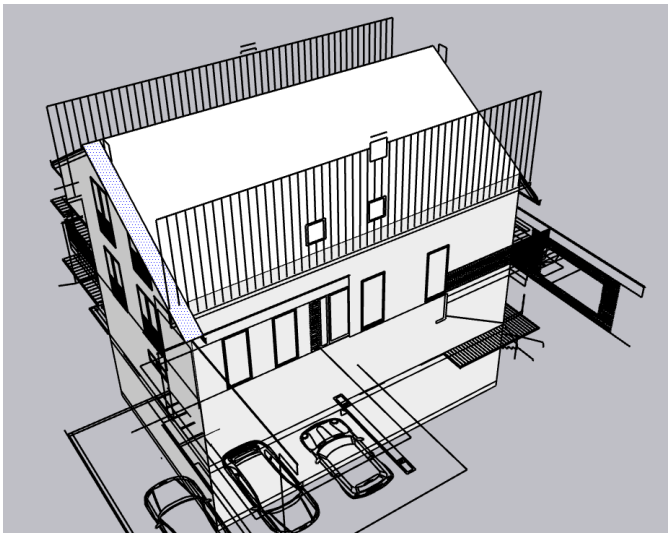


Abbildung 16: Zeichnen des Dachüberstandes aus den CAD-Zeichnungen zweier angrenzender Ansichten.

1.11 Erstellung der Balkone

Wie auch schon bei der Erstellung der Gebäudehülle und der Garage wird für jeden Balkon eine Seite der Balkonplattform nachgezeichnet und auf die entsprechende Länge beziehungsweise Breite des Balkons gemäß der CAD-Zeichnung extrudiert.

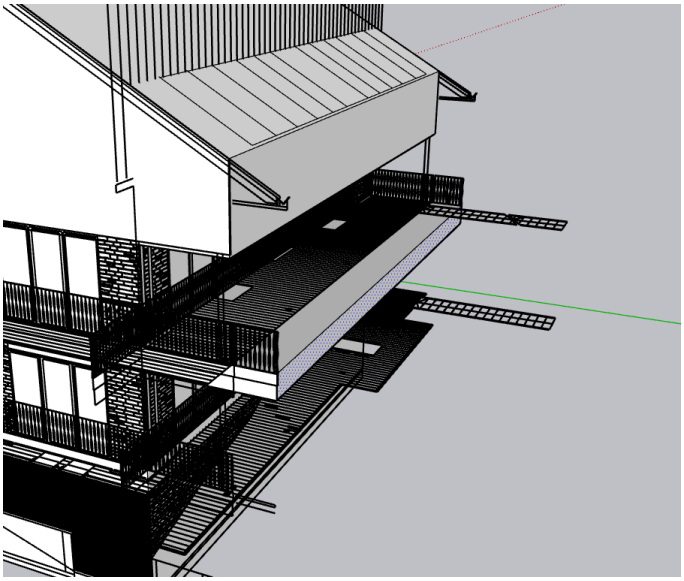


Abbildung 17: Generierung der Balkonplattform.

Anschließend werden die Balkonpfeiler erstellt. Da die Pfeiler wiederkehrende Objekte sind, wird ein Referenzpfeiler als Komponente erstellt, der dann an die jeweiligen Positionen kopiert werden kann. Für die Balkongeländer werden einfach mittels des Linien-Werkzeugs die Endpunkte der CAD-Zeichnungen derart verbunden, dass die jeweilige Geländer-Fläche entsteht. Jeder Balkon wird als individuelles Objekt gruppiert.

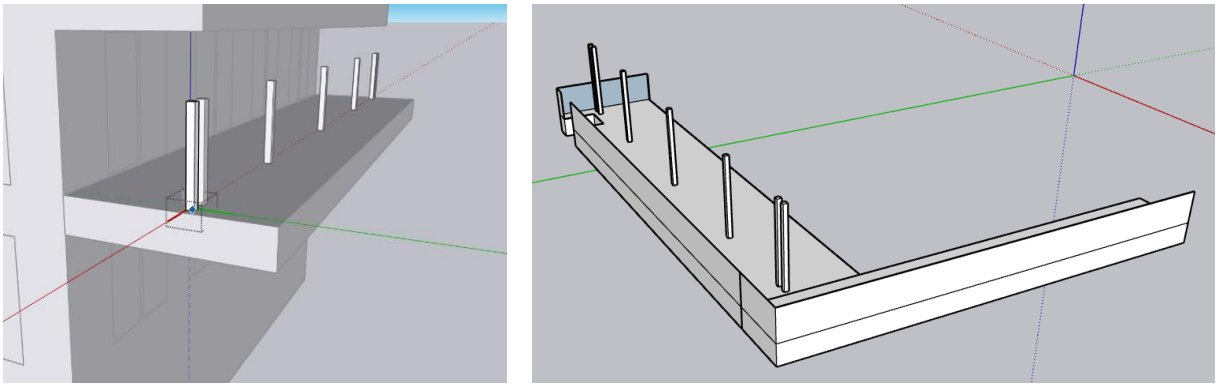


Abbildung 18: Erstellung der Balkonpfeiler und der Geländer.

1.12 Erstellung der Treppe

Die Tiefe der Treppenstufen wird aus einer Ansicht orthogonal zur Treppenrichtung gewonnen (siehe Abbildung 19). Die Breite der Treppe ist einer der Stockwerkszeichnungen zu entnehmen (siehe Abbildung 19). Um nicht jede Treppenstufe einzeln zeichnen zu müssen, wird aus der ersten Treppenstufe eine Komponente erstellt, welche dann beliebig oft kopiert und an die jeweilige Position verschoben wird. Am Ende werden alle Treppenstufen zu einem Objekt gruppiert.

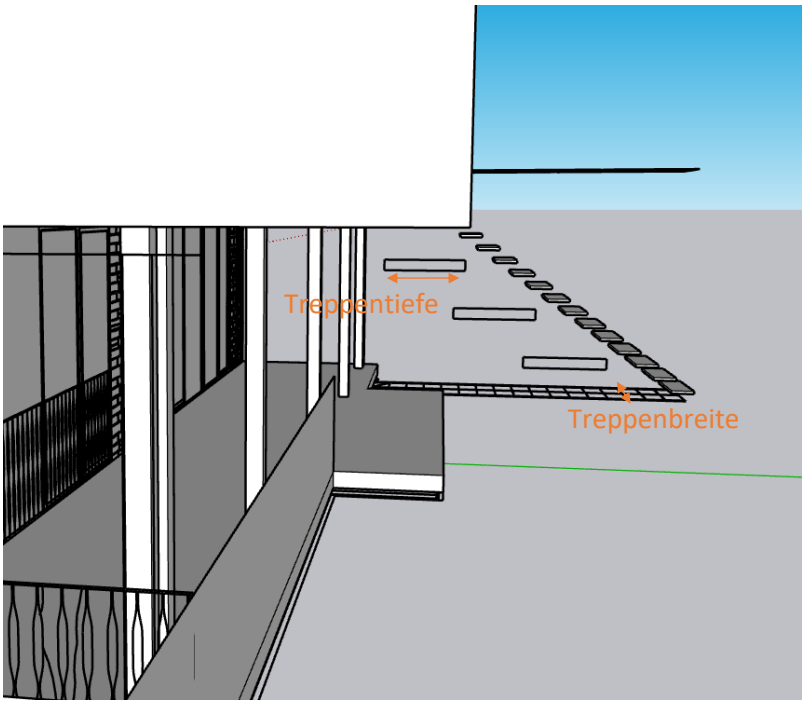


Abbildung 19: Ableitung der Maße der Treppenstufen aus zwei Ansichten.

1.13 SketchUp-Modell

Das Ergebnis des ersten Arbeitsschritts ist ein 3D-Gebäude als SketchUp-Modell, welches aus den 2D-Ansichten einer CAD-Zeichnung abgeleitet wurde.

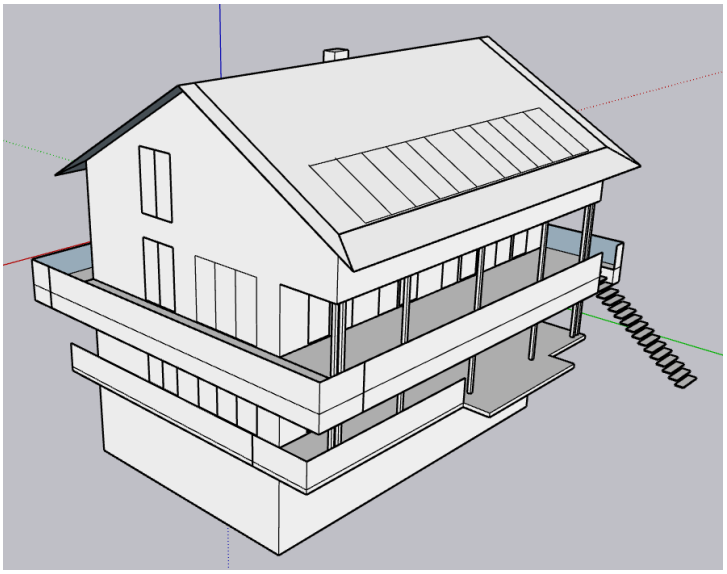


Abbildung 20:Das Resultat des ersten Arbeitsschritts ist ein 3D-SketchUp-Modell des CAD-Gebäudeentwurfs.

2 Generierung eines CityGML-Gebäudemodells

2.1 Instanziierung des CityGML-Gebäudes

Dem Handbuch des GEORES SketchUp CityGML Plugins ist zu entnehmen, dass als erstes eine Gebäude-Instanz erstellt werden muss. Dafür wird die Gebäudehülle selektiert und der gewünschten LoD-Geometrie zugewiesen.

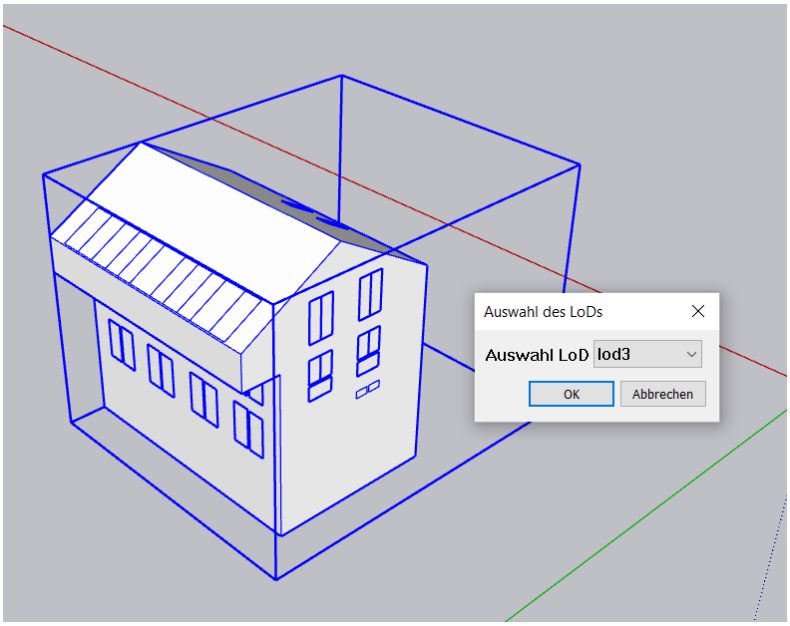


Abbildung 21: Die Gruppierung der Gebäudehülle wird als CityGML-Gebäude instanziiert.

2.2 Zuweisung von thematischen Begrenzungsflächen

Nun werden die Flächen der Gebäudehülle thematischen CityGML-Begrenzungsflächen zugewiesen. Dafür wird die entsprechende Fläche selektiert und anschließend auf den Knopf in der Toolbar (siehe rotes Rechteck in Abbildung 22) für die Zuweisung zur jeweiligen thematischen Fläche geklickt. Dies wird für die Dachflächen, die Wandflächen, die äußeren Deckenflächen, die Fenster, die Türen und die Grundfläche der Gebäudehülle durchgeführt.

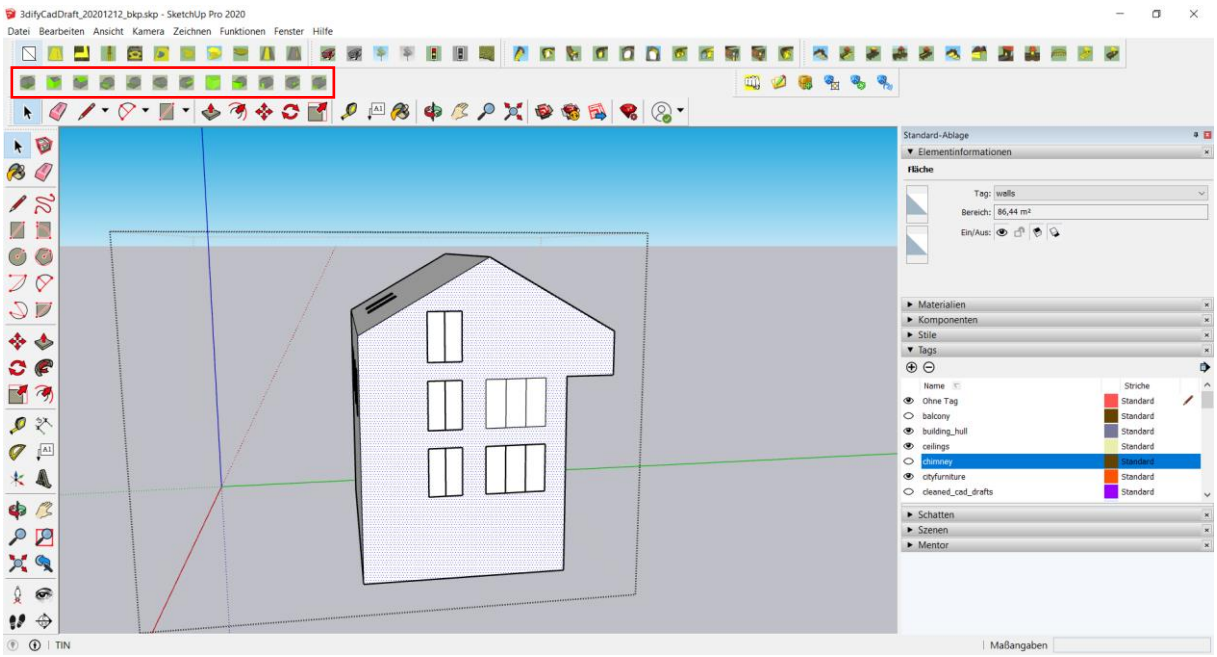


Abbildung 22: Zuweisung einer selektierten Gebäudefläche zu einer thematischen CityGML-Begrenzungsfläche.

2.2 Zuweisung von Gebäudeinstallationen

Nachdem eine Instanz des CityGML-Gebäudes erstellt wurde, können diesem die Gebäudeinstallationen (Dachüberstände, Balkone, Kamin und Treppe) zugewiesen werden. Zuerst wird die Gruppierung einer Gebäudeinstallation selektiert. Durch Rechtsklick auf die ausgewählte Gruppierung erscheint ein Menü, in dem auf *GEORES CityGML Manuell zuweisen* geklickt wird.

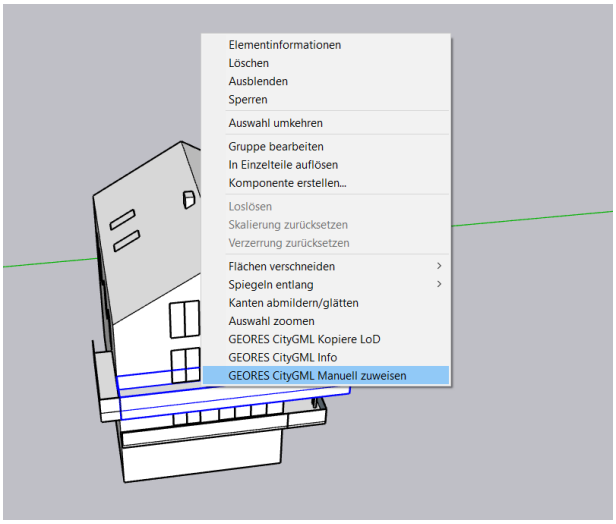


Abbildung 23: Nach Auswahl der Gruppierung der Gebäudeinstallation und Rechtsklick auf diese erscheint das GEORES Plugin Menü.

Nachdem auf den Befehl *GEORES CityGML Manuell zuweisen* geklickt wurde, kann die Gebäudeinstallation durch Klicken auf das Gebäude, welches dann gelb angezeigt wird, diesem zugewiesen werden. Es öffnet sich ein Menü, aus dem nochmal der Name der Gebäudeinstanz ausgewählt werden muss.

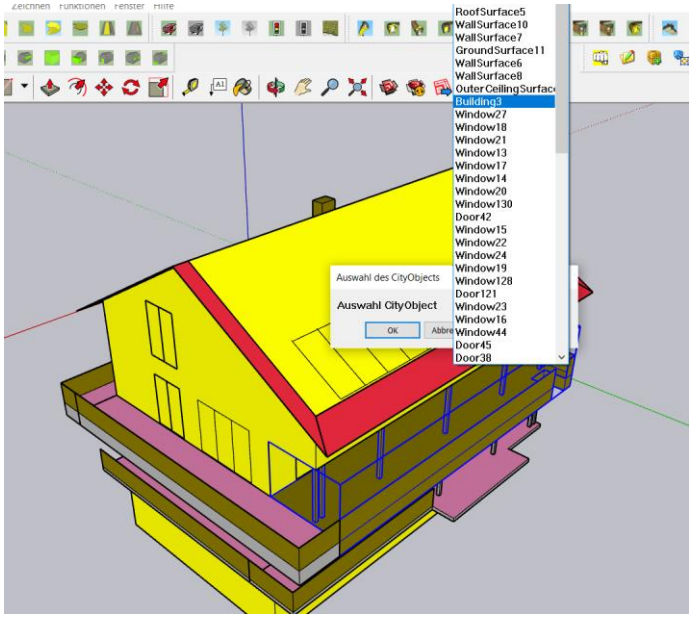


Abbildung 24: Das selektierte Element, zu welchem die Gebäudeinstallation zugewiesen werden soll, wird in gelb angezeigt.

Nun wird die Gruppierung der Gebäudeinstallation erneut ausgewählt und über Klicken auf den Knopf zur Erzeugung einer CityGML-Gebäudeinstallation eine Instanz erzeugt. Anschließend können einzelne Flächen der Gebäudeinstallation selektiert werden und einer thematischen CityGML-Begrenzungsfläche zugewiesen werden. Sollen mehrere Flächen Teil einer gemeinsamen Instanz einer thematischen Begrenzungsfläche sein, so müssen alle betroffenen Flächen selektiert sein, bevor der entsprechende Knopf in der Toolbar geklickt wird. Das beschriebene Vorgehen wird für alle vorhandenen Gebäudeinstallationen angewandt.

2.2 Zuweisung von Gebäudeteilen

Das Vorgehen ist analog zu den Schritten in *2.2 Zuweisung von Gebäudeinstallationen*. Zunächst wird die Gruppierung der Garage der Gebäudeinstanz zugewiesen. Anschließend wird durch Drücken des CityGML-Gebäudeteil-Knopfes in der Toolbar ein CityGML-Gebäudeteil erstellt. Flächen des Gebäudeteils können durch Auswahl und Drücken des entsprechenden Knopfes in der Toolbar einer der zur Verfügung stehenden thematischen CityGML-Begrenzungsflächen zugewiesen werden.

2.4 Export des CityGML-Gebäudemodells

Über die die Auswahl *Funktionen* in der Menüleiste kann zum Befehl *GEORES CityGML Export 2.0* gelangt werden. Durch Klicken auf diesen Befehl öffnet sich ein Menü, in welchem unter anderem die CityGML-Version und der Speicherort des Exports des CityGML-Gebäudemodells einstellbar sind.

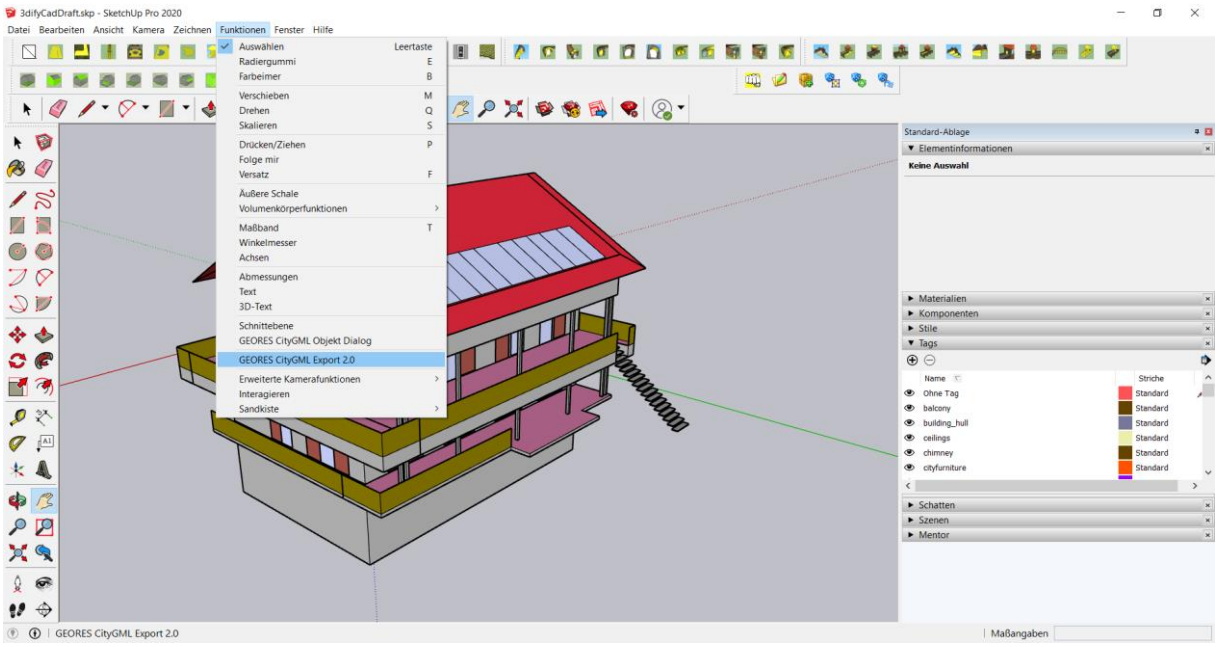


Abbildung 26: Befehl für den Export des SketchUP-Modells im CityGML-Format.

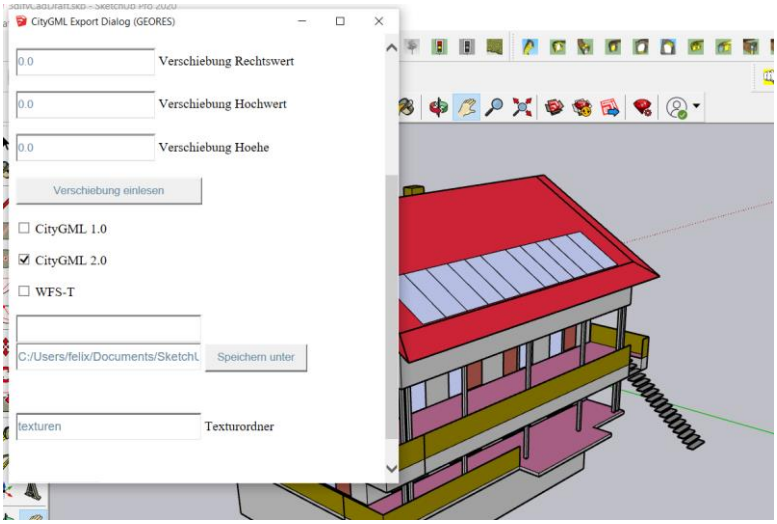


Abbildung 25: Export-Menü des GEORES SketchUp CityGML Plugins.

Zur Überprüfung der exportierten Datei kann diese in einen GML-Viewer, wie den FZKViewer (https://www.iai.kit.edu/1302.php) geladen werden.

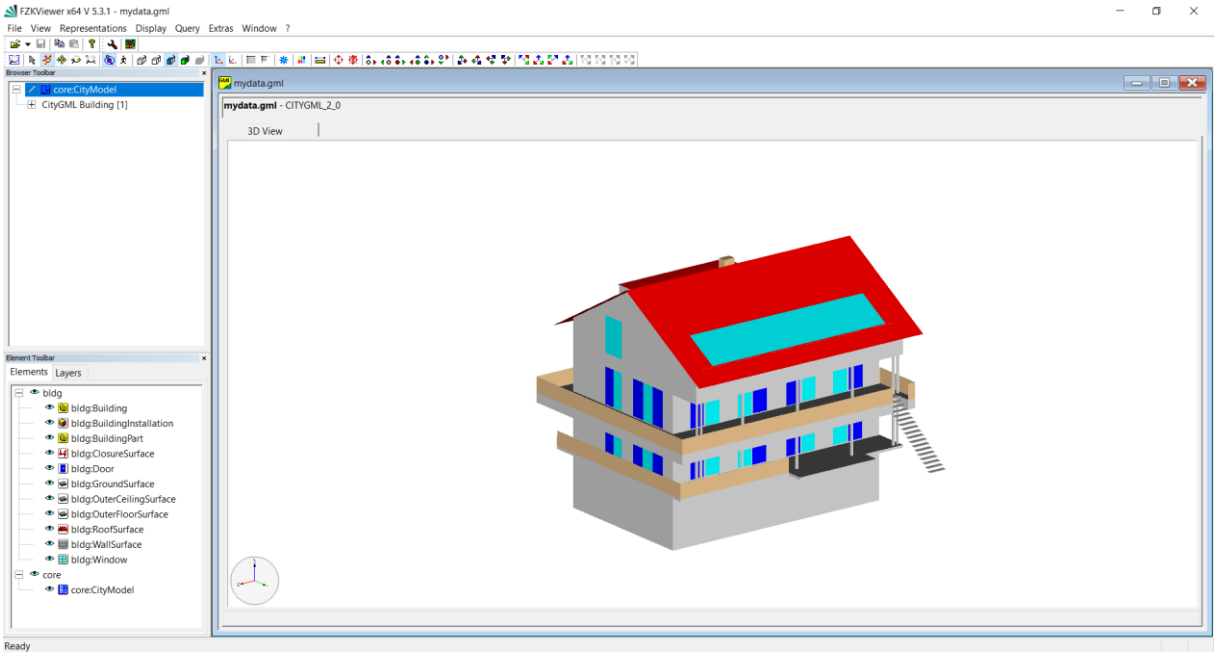
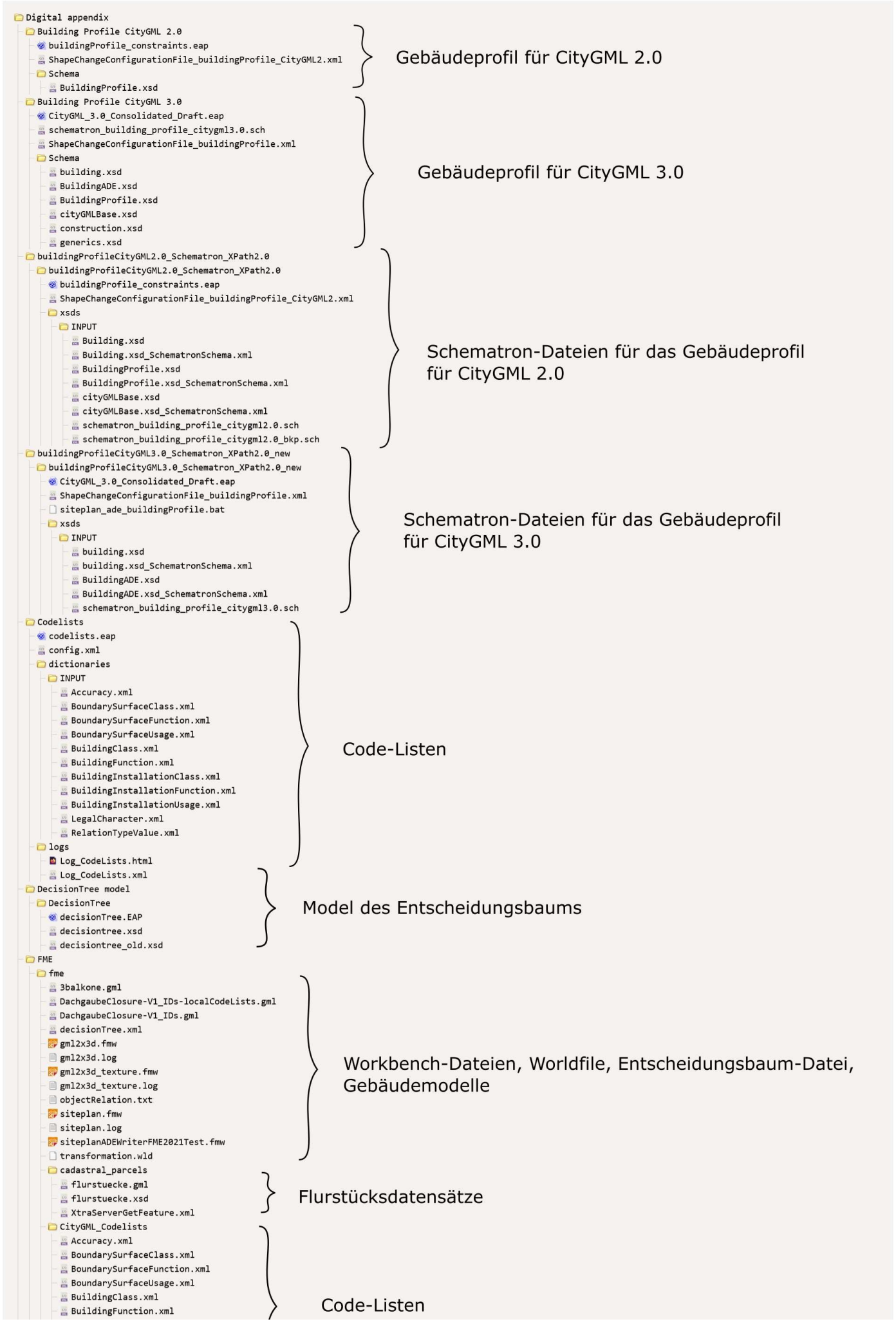
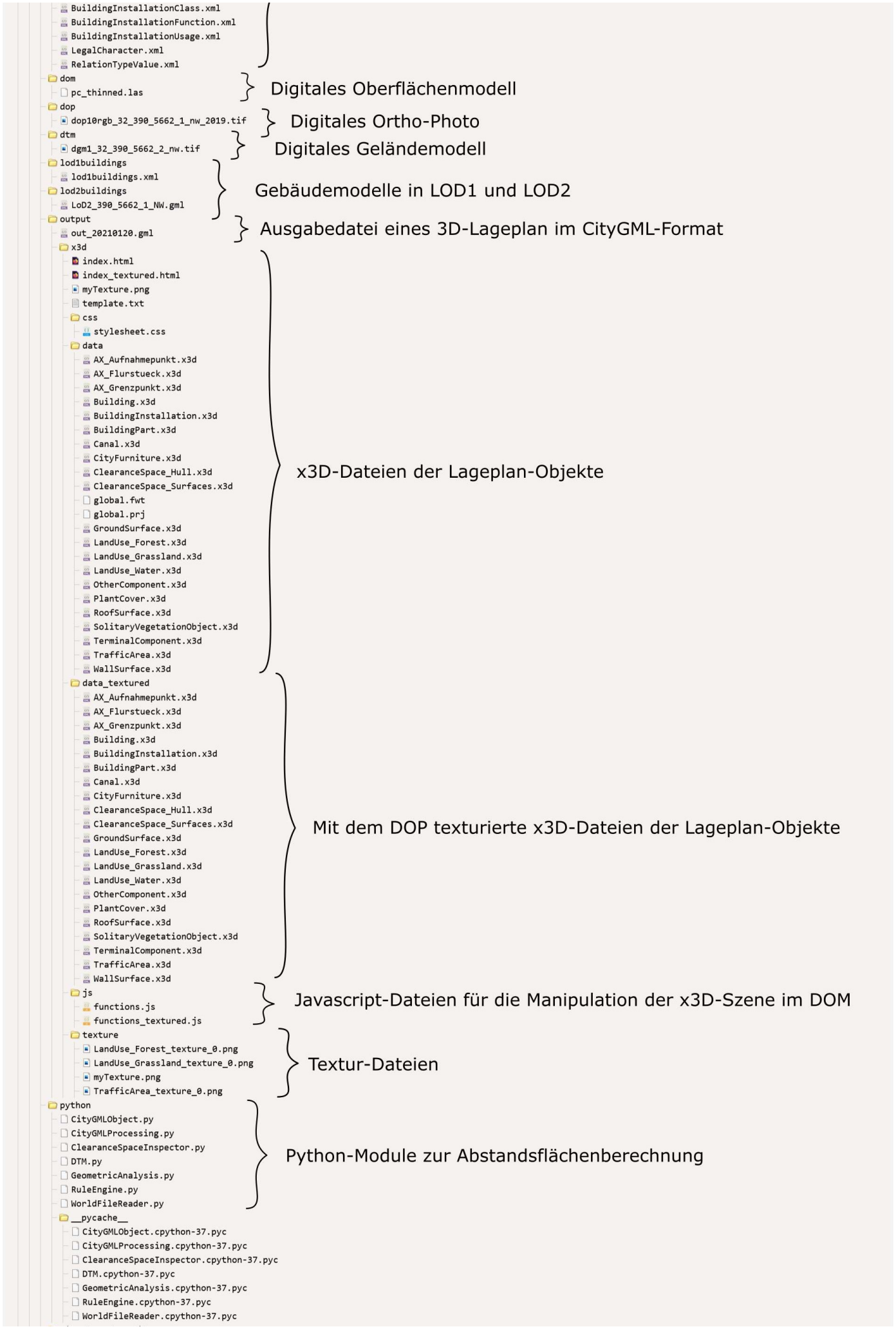
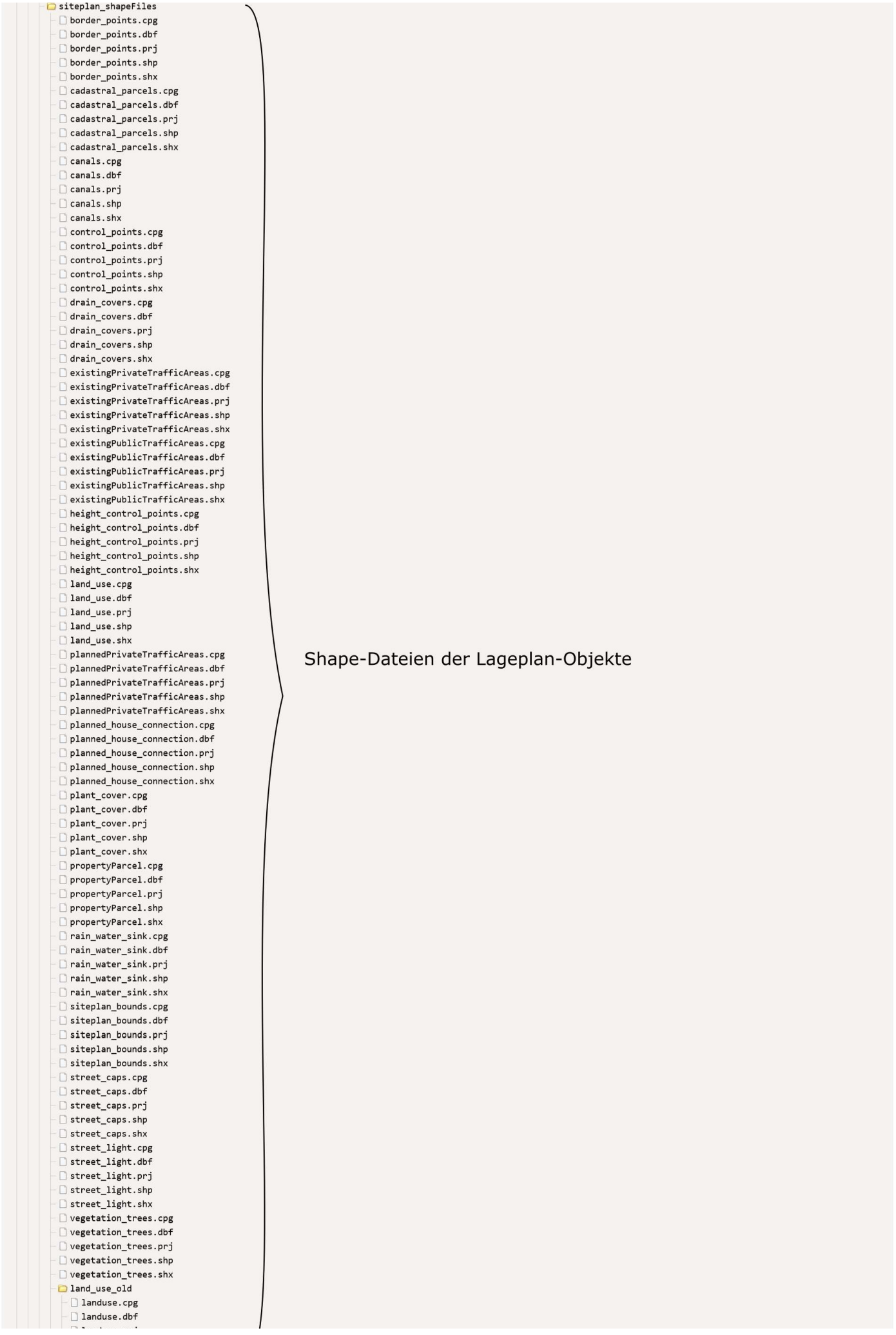


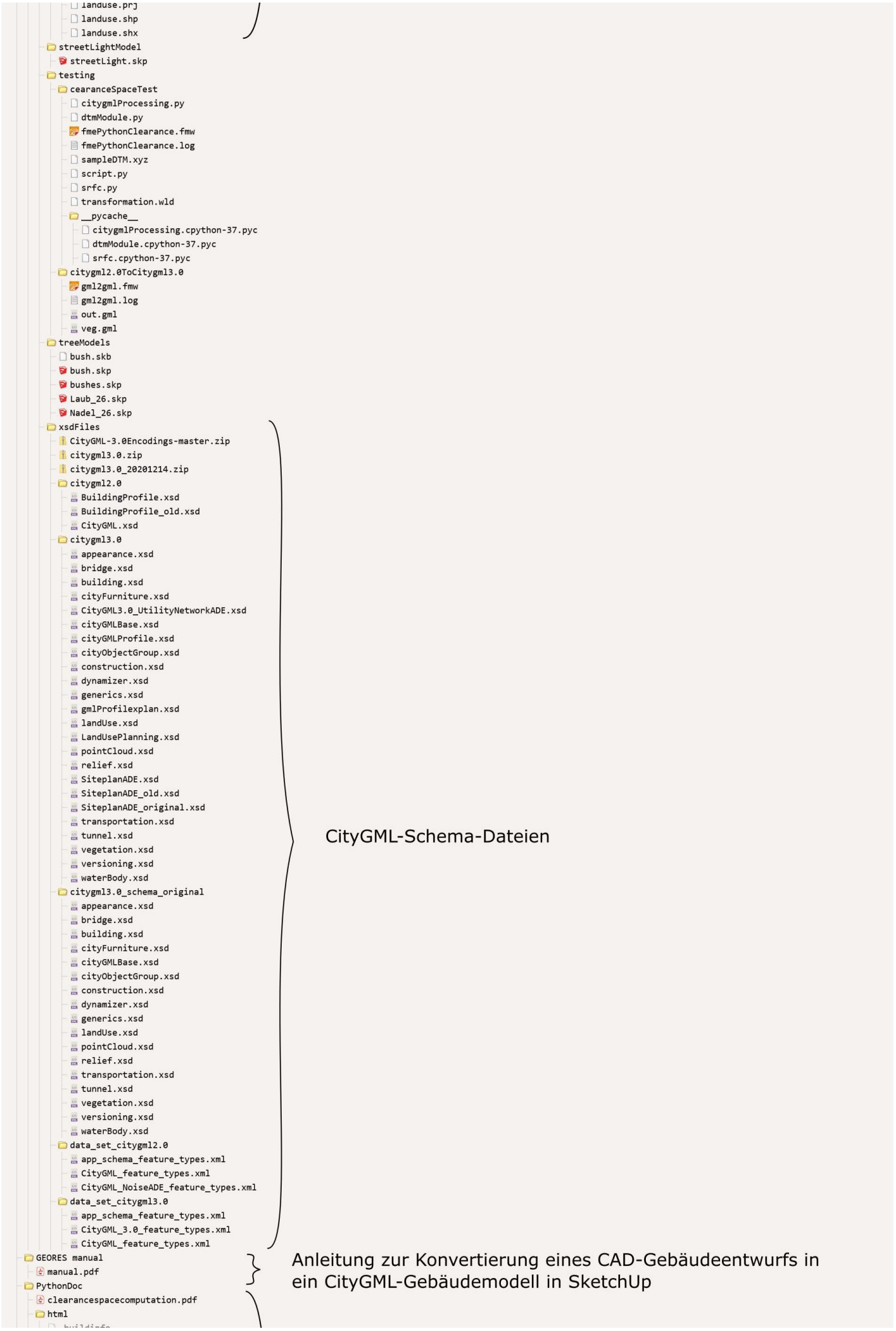
Abbildung 27: Überprüfen des exportierten CityGML-Gebäudemodells im FZKViewer.

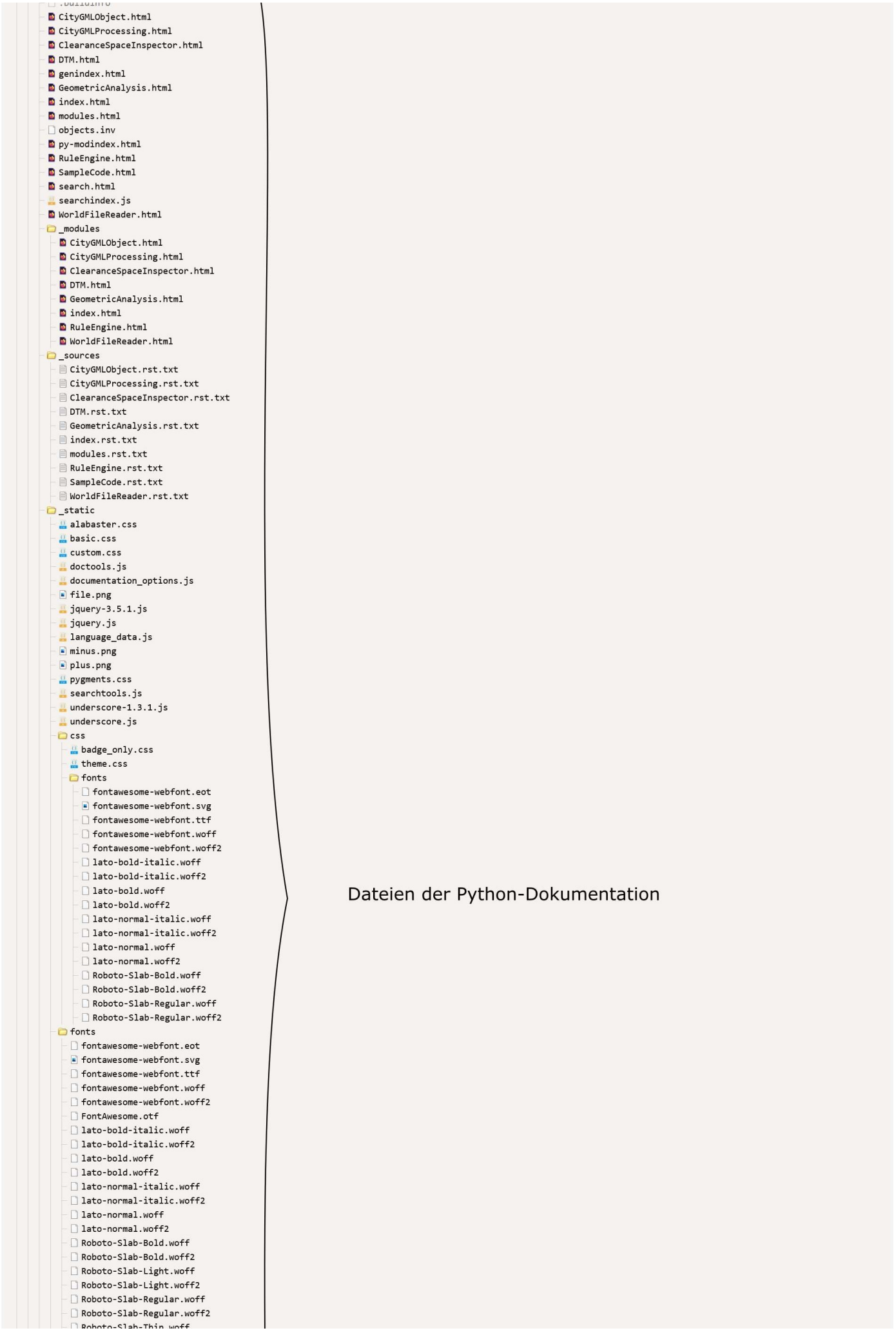
F Verzeichnis des digitalen Anhangs


Digital appendix
Building Profile CityGML 2.0
buildingProfile_constraints.eap
ShapeChangeConfigurationFile_buildingProfile_CityGML2.xml
Schema
BuildingProfile.xsd
Gebäudeprofil für CityGML 2.0
Building Profile CityGML 3.0
CityGML_3.0_Consolidated_Draft.eap
schematron_building_profile_citygml3.0.sch
ShapeChangeConfigurationFile_buildingProfile.xml
Schema
building.xsd
BuildingADE.xsd
BuildingProfile.xsd
cityGMLBase.xsd
construction.xsd
generics.xsd
Gebäudeprofil für CityGML 3.0
buildingProfileCityGML2.0_Schematron_XPath2.0
buildingProfileCityGML2.0_Schematron_XPath2.0
buildingProfile_constraints.eap
ShapeChangeConfigurationFile_buildingProfile_CityGML2.xml
xsds
INPUT
Building.xsd
Building.xsd_SchematronSchema.xml
BuildingProfile.xsd
BuildingProfile.xsd_SchematronSchema.xml
cityGMLBase.xsd
cityGMLBase.xsd_SchematronSchema.xml
schematron_building_profile_citygml2.0.sch
schematron_building_profile_citygml2.0_bkp.sch
Schematron-Dateien für das Gebäudeprofil
für CityGML 2.0
buildingProfileCityGML3.0_Schematron_XPath2.0_new
buildingProfileCityGML3.0_Schematron_XPath2.0_new
CityGML_3.0_Consolidated_Draft.eap
ShapeChangeConfigurationFile_buildingProfile.xml
siteplan_ade_buildingProfile.bat
xsds
INPUT
building.xsd
building.xsd_SchematronSchema.xml
BuildingADE.xsd
BuildingADE.xsd_SchematronSchema.xml
schematron_building_profile_citygml3.0.sch
Schematron-Dateien für das Gebäudeprofil
für CityGML 3.0
Codelists
codelists.eap
config.xml
dictionaries
INPUT
Accuracy.xml
BoundarySurfaceClass.xml
BoundarySurfaceFunction.xml
BoundarySurfaceUsage.xml
BuildingClass.xml
BuildingFunction.xml
BuildingInstallationClass.xml
BuildingInstallationFunction.xml
BuildingInstallationUsage.xml
LegalCharacter.xml
RelationTypeValue.xml
logs
Log_CodeLists.html
Log_CodeLists.xml
Code-Listen
DecisionTree model
DecisionTree
decisionTree.EAP
decisiontree.xsd
decisiontree_old.xsd
Model des Entscheidungsbaums
FME
fme
3balkone.gml
DachgaubeClosure-V1_IDs-localCodeLists.gml
DachgaubeClosure-V1_IDs.gml
decisionTree.xml
gml2x3d.fmw
gml2x3d.log
gml2x3d_texture.fmw
gml2x3d_texture.log
objectRelation.txt
siteplan.fmw
siteplan.log
siteplanADEWriterFME2021Test.fmw
transformation.wld
Workbench-Dateien, Worldfile, Entscheidungsbaum-Datei,
Gebäudemodelle
cadastral_parcels
flurstuecke.gml
flurstuecke.xsd
XtraServerGetFeature.xml
Flurstücksdatensätze
CityGML_Codelists
Accuracy.xml
BoundarySurfaceClass.xml
BoundarySurfaceFunction.xml
BoundarySurfaceUsage.xml
BuildingClass.xml
BuildingFunction.xml
Code-Listen

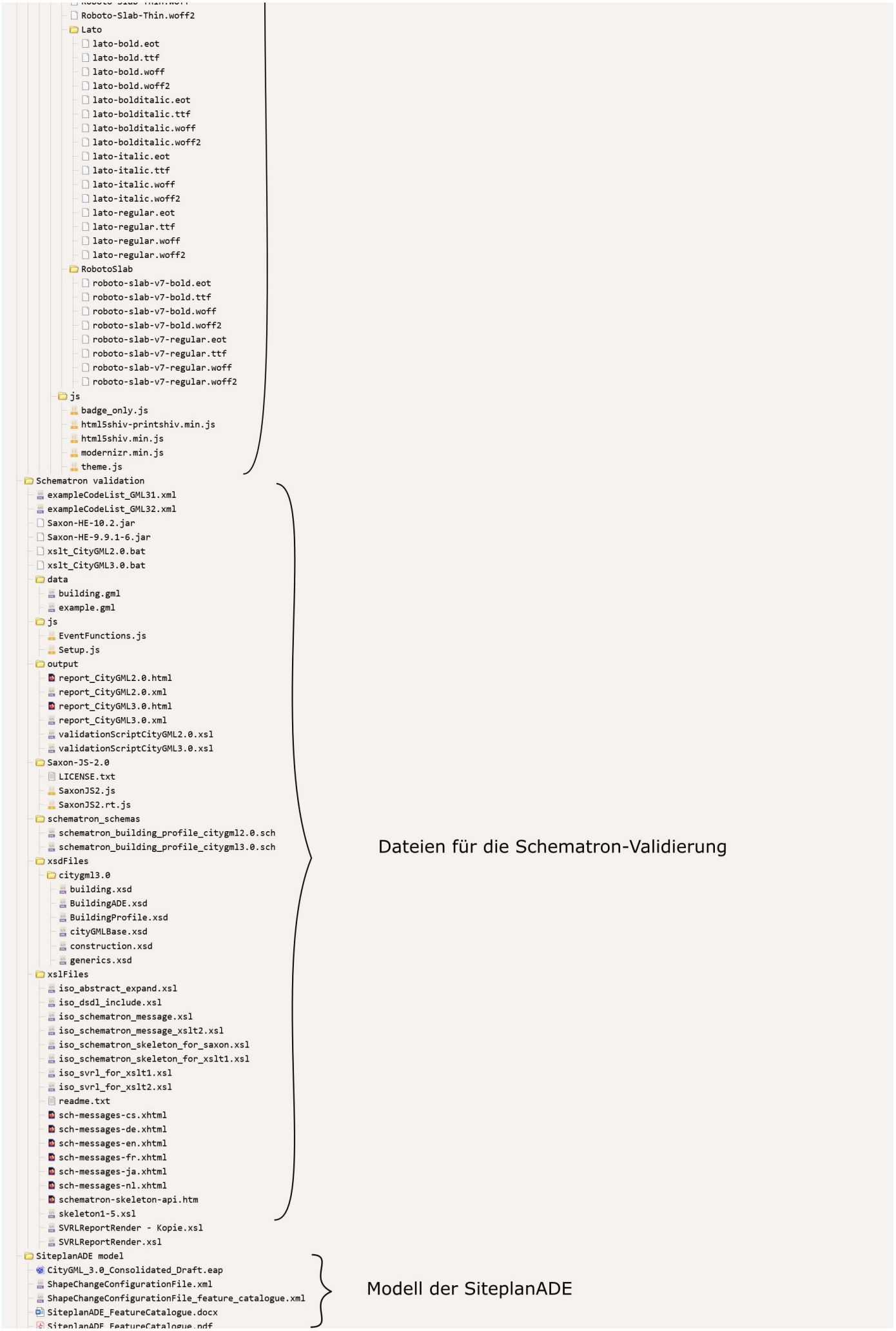

BuildingInstallationClass.xml
BuildingInstallationFunction.xml
BuildingInstallationUsage.xml
LegalCharacter.xml
RelationTypeValue.xml
dom
pc_thinned.las
Digitales Oberflächenmodell
dop
dop10rgb_32_390_5662_1_nw_2019.tif
Digitales Ortho-Photo
dtm
dgm1_32_390_5662_2_nw.tif
Digitales Geländemodell
lod1buildings
lod1buildings.xml
lod2buildings
LoD2_390_5662_1_NW.gml
Gebäudemodelle in LOD1 und LOD2
output
out_20210120.gml
Ausgabedatei eines 3D-Lageplan im CityGML-Format
x3d
index.html
index_textured.html
myTexture.png
template.txt
css
stylesheet.css
data
AX_Aufnahmepunkt.x3d
AX_Flurstueck.x3d
AX_Grenzpunkt.x3d
Building.x3d
BuildingInstallation.x3d
BuildingPart.x3d
Canal.x3d
CityFurniture.x3d
ClearanceSpace_Hull.x3d
ClearanceSpace_Surfaces.x3d
global.fwt
global.prj
GroundSurface.x3d
LandUse_Forest.x3d
LandUse_Grassland.x3d
LandUse_Water.x3d
OtherComponent.x3d
PlantCover.x3d
RoofSurface.x3d
SolitaryVegetationObject.x3d
TerminalComponent.x3d
TrafficArea.x3d
WallSurface.x3d
x3D-Dateien der Lageplan-Objekte
data_textured
AX_Aufnahmepunkt.x3d
AX_Flurstueck.x3d
AX_Grenzpunkt.x3d
Building.x3d
BuildingInstallation.x3d
BuildingPart.x3d
Canal.x3d
CityFurniture.x3d
ClearanceSpace_Hull.x3d
ClearanceSpace_Surfaces.x3d
GroundSurface.x3d
LandUse_Forest.x3d
LandUse_Grassland.x3d
LandUse_Water.x3d
OtherComponent.x3d
PlantCover.x3d
RoofSurface.x3d
SolitaryVegetationObject.x3d
TerminalComponent.x3d
TrafficArea.x3d
WallSurface.x3d
Mit dem DOP texturierte x3D-Dateien der Lageplan-Objekte
js
functions.js
functions_textured.js
Javascript-Dateien für die Manipulation der x3D-Szene im DOM
texture
LandUse_Forest_texture_0.png
LandUse_Grassland_texture_0.png
myTexture.png
TrafficArea_texture_0.png
Textur-Dateien
python
CityGMLObject.py
CityGMLProcessing.py
ClearanceSpaceInspector.py
DTM.py
GeometricAnalysis.py
RuleEngine.py
WorldFileReader.py
Python-Module zur Abstandsflächenberechnung
__pycache__
CityGMLObject.cpython-37.pyc
CityGMLProcessing.cpython-37.pyc
ClearanceSpaceInspector.cpython-37.pyc
DTM.cpython-37.pyc
GeometricAnalysis.cpython-37.pyc
RuleEngine.cpython-37.pyc
WorldFileReader.cpython-37.pyc


siteplan_shapeFiles
border_points.cpg
border_points.dbf
border_points.prj
border_points.shp
border_points.shx
cadastral_parcels.cpg
cadastral_parcels.dbf
cadastral_parcels.prj
cadastral_parcels.shp
cadastral_parcels.shx
canals.cpg
canals.dbf
canals.prj
canals.shp
canals.shx
control_points.cpg
control_points.dbf
control_points.prj
control_points.shp
control_points.shx
drain_covers.cpg
drain_covers.dbf
drain_covers.prj
drain_covers.shp
drain_covers.shx
existingPrivateTrafficAreas.cpg
existingPrivateTrafficAreas.dbf
existingPrivateTrafficAreas.prj
existingPrivateTrafficAreas.shp
existingPrivateTrafficAreas.shx
existingPublicTrafficAreas.cpg
existingPublicTrafficAreas.dbf
existingPublicTrafficAreas.prj
existingPublicTrafficAreas.shp
existingPublicTrafficAreas.shx
height_control_points.cpg
height_control_points.dbf
height_control_points.prj
height_control_points.shp
height_control_points.shx
land_use.cpg
land_use.dbf
land_use.prj
land_use.shp
land_use.shx
plannedPrivateTrafficAreas.cpg
plannedPrivateTrafficAreas.dbf
plannedPrivateTrafficAreas.prj
plannedPrivateTrafficAreas.shp
plannedPrivateTrafficAreas.shx
planned_house_connection.cpg
planned_house_connection.dbf
planned_house_connection.prj
planned_house_connection.shp
planned_house_connection.shx
plant_cover.cpg
plant_cover.dbf
plant_cover.prj
plant_cover.shp
plant_cover.shx
propertyParcel.cpg
propertyParcel.dbf
propertyParcel.prj
propertyParcel.shp
propertyParcel.shx
rain_water_sink.cpg
rain_water_sink.dbf
rain_water_sink.prj
rain_water_sink.shp
rain_water_sink.shx
siteplan_bounds.cpg
siteplan_bounds.dbf
siteplan_bounds.prj
siteplan_bounds.shp
siteplan_bounds.shx
street_caps.cpg
street_caps.dbf
street_caps.prj
street_caps.shp
street_caps.shx
street_light.cpg
street_light.dbf
street_light.prj
street_light.shp
street_light.shx
vegetation_trees.cpg
vegetation_trees.dbf
vegetation_trees.prj
vegetation_trees.shp
vegetation_trees.shx
land_use_old
landuse.cpg
landuse.dbf
Shape-Dateien der Lageplan-Objekte


landuse.prj
landuse.shp
landuse.shx
streetLightModel
streetLight.skp
testing
cearanceSpaceTest
citygmlProcessing.py
dtmModule.py
fmePythonClearance.fmw
fmePythonClearance.log
sampleDTM.xyz
script.py
srfc.py
transformation.wld
__pycache__
citygmlProcessing.cpython-37.pyc
dtmModule.cpython-37.pyc
srfc.cpython-37.pyc
citygml2.0ToCitygml3.0
gml2gml.fmw
gml2gml.log
out.gml
veg.gml
treeModels
bush.skb
bush.skp
bushes.skp
Laub_26.skp
Nadel_26.skp
xsdFiles
CityGML-3.0Encodings-master.zip
citygml3.0.zip
citygml3.0_20201214.zip
citygml2.0
BuildingProfile.xsd
BuildingProfile_old.xsd
CityGML.xsd
citygml3.0
appearance.xsd
bridge.xsd
building.xsd
cityFurniture.xsd
CityGML3.0_UtilityNetworkADE.xsd
cityGMLBase.xsd
cityGMLProfile.xsd
cityObjectGroup.xsd
construction.xsd
dynamizer.xsd
generics.xsd
gmlProfilexplan.xsd
landUse.xsd
LandUsePlanning.xsd
pointCloud.xsd
relief.xsd
SiteplanADE.xsd
SiteplanADE_old.xsd
SiteplanADE_original.xsd
transportation.xsd
tunnel.xsd
vegetation.xsd
versioning.xsd
waterBody.xsd
citygml3.0_schema_original
appearance.xsd
bridge.xsd
building.xsd
cityFurniture.xsd
cityGMLBase.xsd
cityObjectGroup.xsd
construction.xsd
dynamizer.xsd
generics.xsd
landUse.xsd
pointCloud.xsd
relief.xsd
transportation.xsd
tunnel.xsd
vegetation.xsd
versioning.xsd
waterBody.xsd
data_set_citygml2.0
app_schema_feature_types.xml
CityGML_feature_types.xml
CityGML_NoiseADE_feature_types.xml
data_set_citygml3.0
app_schema_feature_types.xml
CityGML_3.0_feature_types.xml
CityGML_feature_types.xml
GEORES manual
manual.pdf
PythonDoc
clearancespacecomputation.pdf
html
CityGML-Schema-Dateien
Anleitung zur Konvertierung eines CAD-Gebäudeentwurfs in ein CityGML-Gebäudemodell in SketchUp



Dateien der Python-Dokumentation


Roboto-Slab-Thin.woff2
Lato
lato-bold.eot
lato-bold.ttf
lato-bold.woff
lato-bold.woff2
lato-bolditalic.eot
lato-bolditalic.ttf
lato-bolditalic.woff
lato-bolditalic.woff2
lato-italic.eot
lato-italic.ttf
lato-italic.woff
lato-italic.woff2
lato-regular.eot
lato-regular.ttf
lato-regular.woff
lato-regular.woff2
RobotoSlab
roboto-slab-v7-bold.eot
roboto-slab-v7-bold.ttf
roboto-slab-v7-bold.woff
roboto-slab-v7-bold.woff2
roboto-slab-v7-regular.eot
roboto-slab-v7-regular.ttf
roboto-slab-v7-regular.woff
roboto-slab-v7-regular.woff2
js
badge_only.js
html5shiv-printshiv.min.js
html5shiv.min.js
modernizr.min.js
theme.js
Schematron validation
exampleCodeList_GML31.xml
exampleCodeList_GML32.xml
Saxon-HE-10.2.jar
Saxon-HE-9.9.1-6.jar
xslt_CityGML2.0.bat
xslt_CityGML3.0.bat
data
building.gml
example.gml
js
EventFunctions.js
Setup.js
output
report_CityGML2.0.html
report_CityGML2.0.xml
report_CityGML3.0.html
report_CityGML3.0.xml
validationScriptCityGML2.0.xsl
validationScriptCityGML3.0.xsl
Saxon-JS-2.0
LICENSE.txt
SaxonJS2.js
SaxonJS2.rt.js
schematron_schemas
schematron_building_profile_citygml2.0.sch
schematron_building_profile_citygml3.0.sch
xsdFiles
citygml3.0
building.xsd
BuildingADE.xsd
BuildingProfile.xsd
cityGMLBase.xsd
construction.xsd
generics.xsd
xslFiles
iso_abstract_expand.xsl
iso_dsdl_include.xsl
iso_schematron_message.xsl
iso_schematron_message_xslt2.xsl
iso_schematron_skeleton_for_saxon.xsl
iso_schematron_skeleton_for_xslt1.xsl
iso_svrl_for_xslt1.xsl
iso_svrl_for_xslt2.xsl
readme.txt
sch-messages-cs.xhtml
sch-messages-de.xhtml
sch-messages-en.xhtml
sch-messages-fr.xhtml
sch-messages-ja.xhtml
sch-messages-nl.xhtml
schematron-skeleton-api.htm
skeleton1-5.xsl
SVRLReportRender - Kopie.xsl
SVRLReportRender.xsl
SiteplanADE model
CityGML_3.0_Consolidated_Draft.eap
ShapeChangeConfigurationFile.xml
ShapeChangeConfigurationFile_feature_catalogue.xml
SiteplanADE_FeatureCatalogue.docx
SiteplanADE_FeatureCatalogue.pdf
Dateien für die Schematron-Validierung
Modell der SiteplanADE

Literatur

About inspire. (2021). https://inspire.ec.europa.eu/about-inspire/563. Zugriff auf https://inspire.ec.europa.eu/about-inspire/563

AKG Software. (2019). *Grundlagen.* https://www.akgsoftware.de/docs/de/infravision/b56/akgcad/webhelp/topics/koordinaten_r_grundlagen.html. Zugriff auf https://www.akgsoftware.de/docs/de/infravision/b56/akgcad/webhelp/topics/koordinaten_r_grundlagen.html

Albert, J., Bachmann, M. & Hellmeier, A. (2003, April). Erhebungen im Rahmen der Arbeitsgruppe„Anwendungen und Zielgruppen“der SIG3Dim Rahmen der Initiative GDI-NRW. In *Zielgruppen und Anwendungen für Digitale Stadtmodelle und Digitale Geländemodelle.* Zugriff auf https://docplayer.org/20358349-Zielgruppen-und-anwendungen-fuer-digitale-stadtmodelle-und-digitale-gelaendemodelle.html

André Borrmann, C. K., Jakob Beetz & Liebich, T. (2015). Building Information Modeling. In A. Borrmann, M. König, C. Koch & J. Beetz (Hrsg.), *Building Information Modeling* (S. 112–114). Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-05606-3

Aringer, K. (2016). Modellbasierte Transformation von 3D-Gebäudemodellen nach INSPIRE. *zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement* (3/2016), 184–191. doi: 10.12902/zfv-0113-2016

Aringer, K. (2017). Geodäsie in einer digitalen Zukunft Erwartungen aus einer amtlichen Perspektive. In *Mitteilungen* (Bd. 3.2017).

Aumann, G. et al. (Hrsg.). (2016). *Bezugssystemwechsel auf ETRS89/UTM - Grundlagen, Erfahrungen undEmpfehlungen.* Andreas Donaubauer, Thomas H. Kolbe. Zugriff auf www.rundertischgis.de/publikationen/leitfaeden

Balka, K., Daub, M. & Pflanzer, A. (2018). *Automatisierung im öffentlichen Sektor* (Studie). McKinsey&Company. Zugriff auf https://www.mckinsey.de/~/media/mckinsey/locations/europeandmiddleeast/deutschland/publikationen/automatisierungimoeffentlichensektor/automatisierungimoeffentlichensektor.pdf

Basic Information. (2015). http://www.citygmlwiki.org/index.php/Basic_Information. Zugriff auf http://www.citygmlwiki.org/index.php/Basic_Information

BAUO NRW 2018: HANDLUNGSEMPFEHLUNG AUF DER GRUNDLAGE DER DIENSTBESPRECHUNGEN MIT DEN BAUAUF-SICHTSBEHÖRDEN IM OKTOBER/NOVEMBER 2018 [Software-Handbuch]. (2018).

Bayerische Vermessungsverwaltung. (2009). *UTM - Abbildung und UTM - Koordinaten.* https://www.ldbv.bayern.de/file/pdf/1910/UTM-Abbildung-und-Koordinaten.pdf. Zugriff auf https://www.ldbv.bayern.de/file/pdf/1910/UTM-Abbildung-und-Koordinaten.pdf

Becker, R., Clemen, C. & Wunderlich, T. (2019). BIM in der Ingenieurvermessung. In *Leitfaden Geodäsie und BIM.* Runder Tisch GIS E.V.

Bitzer, F. & Brisch, K. M. (1999). *Digitale Signatur*. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-60102-6

Brenner, J. (2019, Juli). XPlanung Struktur und Konzepte..

Brüggemann, T. & von Both, P. (2015). 3D-Stadtmodellierung: CityGML. In *Building Information Modeling* (S. 177–192). Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-05606-3_10

Bydłosz, J., Bieda, A. & Parzych, P. (2018, apr). The Implementation of Spatial Planning Objects in a 3D Cadastral Model. *ISPRS International Journal of Geo-Information*, *7* (4), 153. doi: 10.3390/ijgi7040153

Carsten Sommer, A. R. M. A. A. H. W. L., Elena Mucha. (2016, Dezember). *Umwelt- und Kostenvorteile ausgewählter innovativer Mobilitäts- und Verkehrskonzepte im städtischen Personenverkehr Endbericht - Langfassung* (resreport Nr. 3712 96 101). Umweltbundesamt. Zugriff auf https://www.umweltbundesamt.de/sites/default/files/medien/377/publikationen/2016-12-14_umkomoko_endbericht1_fin.pdf

Chaturvedi, K., Smyth, C. S., Gesquière, G., Kutzner, T. & Kolbe, T. H. (2016, oct). Managing Versions and History Within Semantic 3D City Models for the Next Generation of CityGML. In *Advances in 3d geoinformation* (S. 191–206). Springer International Publishing. doi: 10.1007/978-3-319-25691-7_11

Continental Sachversicherung AG. (2005). *Gutachten zur Ermittlung der Versicherungssummen für die Gebäude-Versischerung.* https://www.mybca.de/sites/bca_neu/files/-media/Produktmatrix_Komposit/Ingenieurservice_Muster-Gutachten.pdf. Zugriff auf https://www.mybca.de/sites/bca_neu/files/media/Produktmatrix_Komposit/Ingenieurservice_Muster-Gutachten.pdf

Cruellas, J. C., Karlinger, G., Pinkas, D. & Ross, J. (2003, Februar). *XML Advanced Electronic Signatures (XAdES)* (W3C Note). W3C. Zugriff auf https://www.w3.org/TR/XAdES/

D2.8.I.6 Data Specification on Cadastral Parcels – Technical Guidelines (techreport Nr. D2.8.I.6_v3.1). (2014). European Commission Joint Research Centre. Zugriff auf https://inspire.ec.europa.eu/id/document/tg/cp (This document describes the INSPIRE Data Specification for the spatial data theme CadastralParcels)

D2.8.III.12 Data Specification on Natural Risk Zones – Technical Guidelines (techreport Nr. D2.8.III.12_v3.0). (2013). European Commission Joint Research Centre. Zugriff auf https://inspire.ec.europa.eu/id/document/tg/nz (This document describes the INSPIRE Data Specification for the spatial data theme Natural Risk Zones)

D2.8.III.3 INSPIRE Data Specification on Soil – Draft Guidelines (techreport Nr. D2.8.III.3_-v3.0). (2013). European Commission Joint Research Centre. Zugriff auf https://inspire.ec.europa.eu/id/document/tg/so (This document describes the INSPIRE Data Specification for the spatial data theme Soil)

D2.8.III.4 Data Specification on Land Use - Technical Guidelines (techreport Nr. D2.8.III.4_-v3.0). (2013). European Commission Joint Research Centre. Zugriff auf https://inspire.ec.europa.eu/id/document/tg/lu (This document describes the INSPIRE Data Specification for the spatial data themeLand Use)

de Laat, R. & van Berlo, L. (2010, jan). Integration of BIM and GIS: The development of the CityGML GeoBIM extension. In *Lecture Notes in Geoinformation and Cartography* (S. 211–225). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-12670-3_13

Die Eigenschaften der IFC-Objekte: IfcPropertySet. (o. J.). http://biblus.accasoftware.com/de/ifcpropertyset-die-eigenschaften-der-ifc-objekte/. Zugriff auf http://biblus.accasoftware.com/de/ifcpropertyset-die-eigenschaften-der-ifc-objekte/

Die Stadt für Morgen: Umweltschonend mobil – lärmarm – grün – kompakt – durchmischt (resreport). (2017, Mai). Umweltbundesamt. Zugriff auf https://www.umweltbundesamt.de/sites/default/files/medien/421/publikationen/20170505_stadt_von_morgen_2_auflage_web.pdf

Ding, L., Zhou, Y. & Akinci, B. (2014, oct). Building Information Modeling (BIM) application framework: The process of expanding from 3D to computable nD. *Automation in Construction*, *46*, 82–93. doi: 10.1016/j.autcon.2014.04.009

Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok). (2018). In (Bd. 7.1).

Donkers, S. (2013). *Automatic generationof CityGML LoD3 building models from IFC models* (mathesis, TU Delft, Department of GIS Technology, OTB Research Institute for the Built Environment). Zugriff auf https://repository.tudelft.nl/islandora/object/uuid:31380219-f8e8-4c66-a2dc-548c3680bb8d

Donkers, S., Ledoux, H., Zhao, J. & Stoter, J. (2015, sep). Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings. *Transactions in GIS*, *20* (4), 547–569. doi: 10.1111/tgis.12162

Duan, X. & Benner, J. (2019, Dezember). *Transformationsregeln XPlanung 5.2 nach INSPIRE PLU 4.0* (Version 2.2 Aufl.). http://www.xplanungwiki.de/upload/INSPIRE/XPlanGML_5_2/INSPIRE_Transformation_2_2_2019-12-17.pdf. Zugriff auf http://www.xplanungwiki.de/upload/INSPIRE/XPlanGML_5_2/INSPIRE_Transformation_2_2_2019-12-17.pdf

Effkemann, C. (2019). Anwendung von 3D-Laserscanning und Photogrammetrie zur as-built-Dokumentation von Gebäuden. In *Leitfaden Geodäsie und BIM* (S. 141–142). Runder Tisch GIS E.V.

Egger, F. (2019). 3D-CityGML-Stadtmodelle als Planungsinstrument für BIM-Infrastrukturprojekte am Beispiel der U4-Netzerweiterung auf die Horner Geest in Hamburg. In *Leitfaden Geodäsie und BIM.* Runder Tisch GIS E.V.

Eigentumsgrenzen im Liegenschaftskataster - Hinweise für Grundstückseigentümer (Bericht). (2017). Landesamt für Digitalisierung, Breitband und Vermessung Bayern. https://www.ldbv.bayern.de/file/pdf/12443/5040_Qualifizierte_Antragsvorbereitung_Tipps_Grenzermittlung. Zugriff auf https://www.ldbv.bayern.de/file/pdf/12443/5040_Qualifizierte_Antragsvorbereitung_Tipps_Grenzermittlung

Eine Informationsbroschüre der Bayerischen Vermessungsverwaltung zur Einführung von ALKIS in Bayern. (2013). In *ALKIS kompakt.* Landesamt für Vermessung und Geoinformation in Zusammenarbeit mit dem Bayerischen Staatsministerium der Finanzen. Zugriff auf https://www.ldbv.bayern.de/file/pdf/6155/ALKIS_kompakt_Web_A4.pdf

Elektronik-Kompendium.de. (2021). *Asymmetrische Kryptografie (Verschlüsselung).*

https://www.elektronik-kompendium.de/sites/net/1910111.htm. Zugriff auf https://www.elektronik-kompendium.de/sites/net/1910111.htm

El-Mekawy, M. & Östman, A. (2010). SEMANTIC MAPPING: AN ONTOLOGY ENGINEERINGMETHOD FOR INTEGRATING BUILDING MODELS IN IFC AND CITYGML. In *3rd ISDE DIGITAL EARTH SUMMIT.*

El-Mekawy, M., Östman, A. & Shahzad, K. (2010, jan). Towards Interoperating CityGML and IFC Building Models: A Unified Model Based Approach. In *Lecture notes in geoinformation and cartography* (S. 73–93). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-12670-3_5

Emamgholian, S., Pouliot, J. & Shojaei, D. (2020, sep). MODELLING LAND-USE REGULATION CONFLICTS WITH 3d COMPONENTS TO SUPPORT ISSUING a BUILDING PERMIT. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIV-4/W1-2020*, 41–48. doi: 10.5194/isprs-archives-xliv-4-w1-2020-41-2020

Erbstößer, A.-C. (2019). *Smart Buildings im Internet der Dinge* (resreport). Technologiestiftung Berlin.

Floros, G. S., Ellul, C. & Dimopoulou, E. (2018, sep). INVESTIGATING INTEROPERABILITY CAPABILITIES BETWEEN IFC AND CITYGML LOD 4 – RETAINING SEMANTIC INFORMATION. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4/W10*, 33–40. doi: 10.5194/isprs-archives-xlii-4-w10-33-2018

Garde, J., Jansen, H. & Bläser, D. (2014). Mobilstationen – Bausteine für eine zukunftsfähige Mobilität in der Stadt..

Gebäudeeinmessung zur sicherung des eigentums an grund und boden. (o. J.). https://www.ldbv.bayern.de/vermessung/grundstck/gebaeude.html. Zugriff auf https://www.ldbv.bayern.de/vermessung/grundstck/gebaeude.html

Gernot Steinberg, D. S. & Scherer, J. (2015). Handbuch Mobilstationen Nordrhein-Westfalen [Software-Handbuch]. Zugriff auf https://www.zukunftsnetz-mobilitaet.nrw.de/sites/default/files/downloads/2015-10-14_handbuch_mobilstationen_nrw_download_neu.pdf

Gilbert, T., Rönsdorf, C., Plume, J., Simmons, S., Nisbet, N., Gruler, H.-C., ... Mercer, A. (2020). *Built environment data standards and their integration: an analysis of IFC, CityGML and LandInfra* (Bericht). Open Geospatial Consortium; buildingSMART International. Zugriff am 2020-03-02T00:00:00 auf https://portal.ogc.org/files/?artifact_id=92634 (OGC Document 19-091r1, bSI TR1012)

Grenzfeststellung - grenzwiederherstellung oder grenzermittlung? (o. J.). https://www.ldbv.bayern.de/vermessung/grundstck/grenzvermessung.html. Zugriff auf https://www.ldbv.bayern.de/vermessung/grundstck/grenzvermessung.html

Gröger, G. & George, B. (2012). Springer Handbook Of Geographic Information. In W. Kresse & D. M. Danko (Hrsg.), (S. 309). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-72680-7

Gröger, G., Kolbe, T. H., Nagel, C. & Häfele, K.-H. (2012, April). *OGC City Geography Markup*

Language (CityGML) Encoding Standard (OpenGIS Encoding Standard Nr. OGC 12-019). http://www.opengis.net/spec/citygml/2.0. Zugriff auf http://www.opengis.net/spec/citygml/2.0

Guidelines on Real Property Units and Identifiers. (2004). ECONOMIC COMMISSION FOR EUROPE.

Hellmann, R. (2019). Vom Laserscan zum BIM-Modell – Bestandsdaten effizient in BIM-Projekte überführen. In *Leitfaden Geodäsie und BIM* (S. 129–131). Runder Tisch GIS E.V.

Herle, S., Becker, R., Wollenberg, R. & Blankenbach, J. (2020, feb). GIM and BIM. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88* (1), 33–42. doi: 10.1007/s41064-020-00090-4

Heunecke, O. (2017). Planung und Umsetzung von Bauvorhaben mit amtlichen Lage- und Höhenkoordinaten. *zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement* (3/2017), 180–187. doi: 10.12902/zfv-0160-2017

Hühnlein, D. & Korte, U. (2006). Grundlagen der elektronischen Signatur. Bundesamt für Sicherheit in der Informationstechnik. Zugriff auf https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ElekSignatur/esig_pdf.pdf?__blob=publicationFile&v=3

Hinweise zum Umgang mit Verzerrungen bei UTM-Koordinaten (Bericht). (2019). Bayerische Vermessungsverwaltung. Zugriff auf https://www.ldbv.bayern.de/file/pdf/13714/UTM_Reduktionen.pdf

Hoffstein, J., Pipher, J. & Silverman, J. H. (2014). *An introduction to mathematical cryptography*. Springer New York. doi: 10.1007/978-1-4939-1711-2

Indrajit, A., van Loenen, B., Ploeger, H. & van Oosterom, P. (2020, nov). Developing a spatial planning information package in ISO 19152 land administration domain model. *Land Use Policy*, *98*, 104111. doi: 10.1016/j.landusepol.2019.104111

Innovation durch Forschung - Erneuerbare Energien und Energieeffizienz: Projekte und Ergebnisse der Forschungsförderung 2018 (resreport). (2019). Bundesministerium für Wirtschaft und Energie Deutschland. Zugriff auf https://www.bmwi.de/Redaktion/DE/Publikationen/Energie/innovation-durch-forschung-2018.pdf?__blob=publicationFile&v=14

Jaana Remes, B. B. K. L. S. S. G. S. J. M. J. L. A. C., Jonathan Woetzel & von der Tann, V. (2018, Juni). Smart cities: Digital solutions for a more livable future. In L. Renaud (Hrsg.), . McKinsey & Company.

Jones, S. A. & Laquidara-Carr, D. (2018). *Leading the Future of Building: Connecting Design Insight* (Bericht). Dodge Data & Analytics. Zugriff auf https://damassets.autodesk.net/content/dam/autodesk/www/campaigns/You-can-Subscription/aec-smart-market-brief-connecting-design-insight-en.pdf

Kaden, R. (2016, Juli). *Geometrische ReduktionStrecken- und Flächenverzerrung aufgrund von Projektionen in amtliche Koordinatensysteme.*

Kaden, R. & Kolbe, T. H. (2016, Juni). *3D-Stadtmodelle vs. BIM.* https://docplayer.org/73054672-3d-stadtmodelle-vs-bim.html. Zugriff auf https://

docplayer.org/73054672-3d-stadtmodelle-vs-bim.html

Kaden, R., Seuß, R. & Kolbe, T. H. (2019). Gemeinsamkeiten und Unterschiede zu CAD und GIS. In *Leitfaden Geodäsie und BIM.* Runder Tisch GIS E.V.

Kühn, H., Plazek, M., Schuster, F., Czanderle, F. & Peper, B. H. (2018). *Nicht beim Onlinezugang stehen bleiben – Potenziale der Automatisierung nutzen* (Policy Paper). Institut für den öffentlichen Sektor. Zugriff auf https://publicgovernance.de/media/Policy_Paper_Automatisierung.pdf

Kilsedar, C. E., Fissore, F., Pirotti, F. & Brovelli, M. A. (2019, may). EXTRACTION AND VISUALIZATION OF 3d BUILDING MODELS IN URBAN AREAS FOR FLOOD SIMULATION. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W11*, 669–673. doi: 10.5194/isprs-archives-xlii-2-w11-669-2019

Kolbe, T. H. (2012, Juni). *BIM, CityGML, and Related Standardization.* http://www.kolleg.loel.hs-anhalt.de/landschaftsinformatik/fileadmin/user_upload/_temp_/2012/Documentation/Freitag/01_1400_Kolbe_-_BIM__CityGML__and_related_standardization.pdf. Zugriff auf http://www.kolleg.loel.hs-anhalt.de/landschaftsinformatik/fileadmin/user_upload/_temp_/2012/Documentation/Freitag/01_1400_Kolbe_-_BIM__CityGML__and_related_standardization.pdf

Kolbe, T. H. (2019). *Applied Geoinformatics I: CityGML – A GML3 application modelfor 3D city & landscape models.*

Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C. & Heazel, C. (2020, Juni). *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard* (OGC® Standard Nr. 0.9). Zugriff auf http://docs.ogc.org/DRAFTS/20-010.pdf

Krause, K.-U. (o. J.). *Was ist XPlanung?* https://www.xleitstelle.de/xplanung. Zugriff auf https://www.xleitstelle.de/xplanung

Krause, U. (2013, September). *Abstandsflächenrecht.* https://docplayer.org/23455238-Abstandsflaechenrecht-uwe-krause-dipl-ing-oebvi-bdvi-brandenburg-arbeitsgruppe-baurecht.html. Zugriff auf https://docplayer.org/23455238-Abstandsflaechenrecht-uwe-krause-dipl-ing-oebvi-bdvi-brandenburg-arbeitsgruppe-baurecht.html

Kreissparkasse Esslingen-Nürtingen. (2016). *Checkliste - Welche Unterlagen benötige ich für eine Baufinanzierung?* https://www.ksk-es.de/content/dam/myif/ksk-esslingen-nuertingen/work/dokumente/privatkunden/baufinanzierung/checkliste_unterlagen_baufinanzierung_kskes.pdf?n=true. Zugriff auf https://www.ksk-es.de/content/dam/myif/ksk-esslingen-nuertingen/work/dokumente/privatkunden/baufinanzierung/checkliste_unterlagen_baufinanzierung_kskes.pdf?n=true

Kumar, K., Labetski, A., Ohori, K. A., Ledoux, H. & Stoter, J. (2019, jul). The LandInfra standard and its role in solving the BIM-GIS quagmire. *Open Geospatial Data, Software and Standards*, *4* (1). doi: 10.1186/s40965-019-0065-z

Kumar, K., Ledoux, H. & Stoter, J. (2016, oct). A CITYGML EXTENSION FOR HANDLING VERY LARGE TINS. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial*

Information Sciences, *IV-2/W1*, 137–143. doi: 10.5194/isprs-annals-iv-2-w1-137-2016

Kutscherauer, N. (2018). *Schematron quickfixes.* https://www.schematron-quickfix.com/index.html. Zugriff auf https://www.schematron-quickfix.com/index.html

Kutzner, T., Chaturvedi, K. & Kolbe, T. H. (2020, feb). CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88* (1), 43–61. doi: 10.1007/s41064-020-00095-z

Landeshauptstadt München Referat für Statdplanung und Bauordnung Lokalbaukomission. (2017, Dezember). *Der vollständige Bauantrag - Der schnelle Weg zur Baugenehmigung* (7. Aufl.). https://www.muenchen.de/rathaus/dam/jcr:e7476179-f4b5-4b0f-8e8c-2770aa579713/VBA_Auflage7_web_download_aktuell2019.pdf. Zugriff auf https://www.muenchen.de/rathaus/dam/jcr:e7476179-f4b5-4b0f-8e8c-2770aa579713/VBA_Auflage7_web_download_aktuell2019.pdf

Laura Gebhardt, R. O. M. K. G. D. H. M. K., Daniel Krajzewicz & Wagner, P. (2018). *Intemodal Urban Modility: Uers, Uses and Use Cases* (resreport). DLR Germany. Zugriff auf https://verkehrsforschung.dlr.de/public/documents/2018/TRA2016_poster_UrMo_final-Upload.pdf

Laurent, D. (2017). The ladm standard in and out of the inspire box..

Lee, D. T. & Lin, A. K. (1986, sep). Generalized delaunay triangulation for planar graphs. *Discrete & Computational Geometry*, *1* (3), 201–217. doi: 10.1007/bf02187695

Lemmen, C. (2012, 07). A Domain Model for Land Administration. *Journal of Theoretical Biology - J THEOR BIOL*.

Lemmen, C., Oosterom, P., Kara, A., Kalogianni, E., Shnaidnman, A., Indrajit, A. & Alattas, A. (2019, 10). The scope of LADM revision is shaping-up..

Lemmen, C., Oosterom, P. V. & Netherlands, T. (2010). The Modelling of Rights, Restrictions and Responsibilities (RRR) in the Land Administration Domain Model (LADM)..

Lemmen, C., van Oosterom, P. & Bennett, R. (2015, dec). The Land Administration Domain Model. *Land Use Policy*, *49*, 535–545. doi: 10.1016/j.landusepol.2015.01.014

Löwner, P. D.-I. M.-O., Casper, D.-I. E., Benner, D.-I. J., Häfele, D.-I. K.-H., rer. nat. Gerhard Gröger, P.-D. D., Gruber, D. I. U., ... Kaden, M. S. R. (2013). CityGML 2.0 – Ein internationaler Standard für 3D-Stadtmodelle. Teil 2: Praxis. In (138. Jg. 2/2013 Aufl., S. 131–143). zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement.

Maarten De Groote, J. V. (2017). How ready is Europe for the smart building revolution? In *eceee 2017 summer study on energy efficiency: Consumption, efficiency and limits* (S. 789–798).

Maarten De Groote, J. V. & Bean, F. (2017). *IS EUROPE READY FOR THE SMARTBUILDINGS REVOLUTION?* Buildings Performance Institute Europe (BPIE). Zugriff auf http://bpie.eu/wp-content/uploads/2017/02/STATUS-REPORT-Is-Europe-ready_FINAL_LR.pdf

Mekawy, M. (2010). *Integrating BIM and GIS for 3D city modelling : the case of IFC and CityGML*. Stockholm: Kungliga Tekniska högskolan.

Mitchell, J. (2020, Januar). User Guide for Geo-referencing in IFC "How to Setup Geo-

referencing in a Building or Linear Infrastructure Model" [Software-Handbuch]. Zugriff auf https://www.buildingsmart.org/wp-content/uploads/2020/02/User-Guide-for-Geo-referencing-in-IFC-v2.0.pdf

Morvaj, B., Lugaric, L. & Krajcar, S. (2011, 01). Demonstrating smart buildings and smart grid features in a smart energy city.

Nagel, C., Stadler, A. & Kolbe, T. (2009, 01). Conceptual requirements for the automatic reconstruction of building information models from uninterpreted 3D models. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *38*.

Noack, G., Ebert, C., Jakubeit, T., Krause, U., Rodemerk, G., Ruge, W., ... Schultz, W. (2005, September). Kommentar zum Amtlichen Lageplan der Verordnung über Vorlagen und Nachweise in bauaufsichtlichen Verfahren im Land Brandenburg. In *Musterlageplan.* Bund der Öffentlich bestellten Vermessungsingenieure (BDVI) e.V. Zugriff auf https://www.bdvi.de/application/files/7415/6940/3469/MusterAL_2005.pdf

Noardo, F., Ellul, C., Harrie, L., Overland, I., Shariat, M., Ohori, K. A. & Stoter, J. (2019a, jun). Opportunities and challenges for GeoBIM in Europe: developing a building permits use-case to raise awareness and examine technical interoperability challenges. *Journal of Spatial Science*, *65* (2), 209–233. doi: 10.1080/14498596.2019.1627253

Noardo, F., Ellul, C., Harrie, L., Overland, I., Shariat, M., Ohori, K. A. & Stoter, J. (2019b, jun). Opportunities and challenges for GeoBIM in europe: developing a building permits use-case to raise awareness and examine technical interoperability challenges. *Journal of Spatial Science*, *65* (2), 209–233. doi: 10.1080/14498596.2019.1627253

Noardo, F., Malacarne, G., Ventura, S. M., Tagliabue, L. C., Ciribini, A. L. C., Ellul, C., ... Stoter, J. (2020, sep). INTEGRATING EXPERTISES AND AMBITIONS FOR DATA-DRIVEN DIGITAL BUILDING PERMITS – THE EUNET4dbp. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIV-4/W1-2020*, 103–110. doi: 10.5194/isprs-archives-xliv-4-w1-2020-103-2020

Ohori, K. A., Biljecki, F., Kumar, K., Ledoux, H. & Stoter, J. (2018). Modelling Cities and Landscapes in 3D with CityGML. In *Building Information Modeling* (S. 199–215). Springer International Publishing. doi: 10.1007/978-3-319-92862-3_11

Paasch, J. (2012, 01). The Legal Cadastral Domain Model..

Paasch, J., Oosterom, P., Paulsson, J. & Lemmen, C. (2013, 05). Specialization of the Land Administration Domain Model (LADM) -An Option for Expanding the Legal Profiles..

Paasch, J. M., van Oosterom, P., Lemmen, C. & Paulsson, J. (2015, dec). Further modelling of LADM's rights, restrictions and responsibilities (RRRs). *Land Use Policy*, *49*, 680–689. doi: 10.1016/j.landusepol.2014.12.013

Paganini, P. (2013, Mai). *What is a digital signature? Fundamental principles.* https://securityaffairs.co/wordpress/5223/digital-id/what-is-a-digital-signature-fundamental-principles.html. Zugriff auf https://securityaffairs.co/wordpress/5223/digital-id/what-is-a-digital-signature-fundamental-principles.html

Pauly, R. (2019a, Juni). ALKIS-Objektartenkatalog DLKM. In *Objektartenkataloge zur Geo-*

InfoDok (Bd. 7.1.0).

Pauly, R. (2019b, Juni). Objektartenkatalog des AAA-Anwendungsschema. In *Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens* (Bd. 7.1.0, S. pp. 263).

Poellet, I. (2008). https://commons.wikimedia.org/wiki/File:Failing_Office_Building_detail_-_Portland_Oregon.jpg. Zugriff auf https://commons.wikimedia.org/wiki/File:Failing_Office_Building_detail_-_Portland_Oregon.jpg

Raffer, C. (2019). Automatisierung und digitale Assistenzsysteme. In *Public governance - zeitschrift fÜr Öffentliches management* (Bd. Winter 2018/2019). Institut für den öffentlichen Sektor. Zugriff auf https://publicgovernance.de/media/PG_Winter_2018_Schwerpunkt_AutomatisierungunddigitaleAssistenzsysteme.pdf

Rajabifard, A., Atazadeh, B. & Kalantari, M. (2019). BIM and Urban Land Administration. In (S. 127–128). CRC Press.

Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec. (2014). http://data.europa.eu/eli/reg/2014/910/oj. Zugriff auf http://data.europa.eu/eli/reg/2014/910/oj

Robertson, E. (2003, November). *An introduction to schematron.* https://www.xml.com/pub/a/2003/11/12/schematron.html. Zugriff auf https://www.xml.com/pub/a/2003/11/12/schematron.html

Roessler, T., Eastlake, D., Nyström, M., Solo, D., Hirsch, F., Reagle, J. & Yiu, K. (2013, April). *XML Signature Syntax and Processing Version 1.1* (W3C Recommendation). W3C. Zugriff auf https://www.w3.org/TR/xmldsig-core1/

Rumor, M., Coors, V., Fendel, E. M. & Zlatanova, S. (2007). Urban and Regional Data Management: UDMS 2007 Annual. In (S. 164). CRC Press.

R+V Versicherungen. (2015). *Immobilienerwerb: Welche Unterlagen sind nötig?* https://www.ruv.de/static-files/ruvde/Content/ratgeber/bauen-wohnen/finanzierung/kreditunterlagen-finanzierung/Tabelle_Immofinanzierung.pdf. Zugriff auf https://www.ruv.de/static-files/ruvde/Content/ratgeber/bauen-wohnen/finanzierung/kreditunterlagen-finanzierung/Tabelle_Immofinanzierung.pdf

Salheb, N. (2019). *Automatic Conversion of CityGML to IFC* (Unveröffentlichte Diplomarbeit). TU Delft Faculty of the Built Environment & Architecture.

Sananthana, K. (2019, August). *Configure your JAVA program with TOML file.* https://medium.com/swlh/configure-your-java-program-with-toml-file-9f779a6a3de6. Zugriff auf https://medium.com/swlh/configure-your-java-program-with-toml-file-9f779a6a3de6

Schönhut, S. (2018). EIN UNTERGRUND-UMWELTDATENMODELL ZUR INTEGRATION VON BIM- UND 3DGIS-DATEN. In *Workshop "3D-Stadtmodelle" - 14.11.18.*

Schröder, J. (2018a). *Bauvorhaben.* https://www.oebvi-schroeder.de/hilfe/bau.html. Zugriff auf https://www.oebvi-schroeder.de/hilfe/bau.html

Schröder, J. (2018b). *Die Baukontrollmessung.* https://www.oebvi-schroeder.de/leistungen/baukontrolle.html. Zugriff auf https://www.oebvi-schroeder.de/leistungen/baukontrolle.html

Schüttel, M. (2009). AAA-konforme Modellierung von Geofachdaten. In (134. Jg. 1/2009 Aufl., S. 11–21). zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement.

Seifert, M. (2005). Das AFIS-ALKIS-ATKIS-Anwendungsschema als Komponente einer Geodateninfrastruktur. In (130. Jg. 2/2005 Aufl., S. 77–81). zfv – Zeitschrift für Geodäsie, Geoinformation und Landmanagement.

Seifert, M. (2012). Springer Handbook of Geographic Information. In W. Kresse & D. M. Danko (Hrsg.), (S. 657–681). Springer Berlin Heidelberg. doi: 10.1007/978-3-540-72680-7

Siegel, E. (2019). *An Introduction to XProc 3.0.* https://www.xml.com/articles/2019/11/05/introduction-xproc-30/. Zugriff auf https://www.xml.com/articles/2019/11/05/introduction-xproc-30/

Simon, M. (2012). Risikomodellierungauf Basis eines 3D-Gebäudemodells –Kooperationsprojekt zwischender Munich Re und der LMU München. In *Mitteilungen* (Bd. 3/2012, S. 225–232). DVW Bayern e.V. Gesellschaft für Geodäsie, Geoninformation und Landmanagement. Zugriff auf https://bayern.dvw.de/sites/default/files/landesverband/bayern/anhang/beitragskontext/2014/simon.pdf

Software systems that provide CityGML support. (2019, Juni). http://www.citygmlwiki.org/index.php?title=Commercial_Software. Zugriff auf http://www.citygmlwiki.org/index.php?title=Commercial_Software

Special Interest Group 3D. (2018a, April). *Modeling Guide for 3D Objects - Part 1: Basics (Rules for Validating GML Geometries in CityGML).* http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_-1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:Solid. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:Solid

Special Interest Group 3D. (2018b, April). *Modeling Guide for 3D Objects - Part 1: Basics (Rules for Validating GML Geometries in CityGML).* http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_-1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:MultiSurface. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:MultiSurface

Special Interest Group 3D. (2018c, April). *Modeling Guide for 3D Objects - Part 1: Basics (Rules for Validating GML Geometries in CityGML).* http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_-1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:Polygon. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_1:_Basics_(Rules_for_Validating_GML_Geometries_in_CityGML)#gml:Polygon

Special Interest Group 3D. (2018d, April). *Modeling Guide for 3D Objects - Part 2: Modeling of Buildings (LoD1, LoD2, LoD3).* http://en.wiki.quality.sig3d.org/index.php/Modeling_-Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_-LoD3)#Building_.28bldg:Building.29. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Building_.28bldg:Building.29

Special Interest Group 3D. (2018e, April). *Modeling Guide for 3D Objects - Part 2: Modeling of Buildings (LoD1, LoD2, LoD3).* http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Wall_-Surfaces_.28bldg:WallSurface.29. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Wall_Surfaces_.28bldg:WallSurface.29

Special Interest Group 3D. (2018f, April). *Modeling Guide for 3D Objects - Part 2: Modeling of Buildings (LoD1, LoD2, LoD3).* http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Roof_-Surfaces_.28bldg:RoofSurface.29. Zugriff auf http://en.wiki.quality.sig3d.org/index.php/Modeling_Guide_for_3D_Objects_-_Part_2:_Modeling_of_Buildings_(LoD1,_LoD2,_LoD3)#Roof_Surfaces_.28bldg:RoofSurface.29

Stadler, A. & Kolbe, T. H. (2007). Spatio-semantic coherence in the integration of 3D city models. *WG II/75th International SymposiumSpatial Data Quality 2007Theme: Modelling qualities in space and time*. Zugriff auf https://www.isprs.org/proceedings/XXXVI/2-C43/Session1/paper_Stadler.pdf

Stouffs, R., Tauscher, H. & Biljecki, F. (2018, aug). Achieving complete and near-lossless conversion from IFC to CityGML. *ISPRS International Journal of Geo-Information*, *7* (9), 355. doi: 10.3390/ijgi7090355

Strategie automatisiertes und vernetztes Fahren (resreport). (2015). Bundesministerium für Verkehr und digitale Infrastruktur. Zugriff auf https://www.bmvi.de/SharedDocs/DE/Publikationen/DG/broschuere-strategie-automatisiertes-vernetztes-fahren.pdf?__blob=publicationFile

Theiler, M. (2020a, März). *BIM-basierter Bauantrag - Projektergebnisse.* https://bim-bauantrag.blogs.ruhr-uni-bochum.de/wp-content/uploads/2020/04/002_Projektergebnisse.pdf. Zugriff auf https://bim-bauantrag.blogs.ruhr-uni-bochum.de/wp-content/uploads/2020/04/002_Projektergebnisse.pdf

Theiler, M. (2020b, März). *BIM-basierter Bauantrag - Prototypische Software.* https://bim-bauantrag.blogs.ruhr-uni-bochum.de/wp-content/uploads/2020/04/003_Prototypische_Software.pdf. Zugriff auf https://bim-bauantrag.blogs.ruhr-uni-bochum.de/wp-content/uploads/2020/04/003_Prototypische_Software.pdf

Theiler, M., Tulke, J., König, M. & Krause, K.-U. (2019). BIM-basierter Bauantrag +. In *Leitfaden Geodäsie und BIM (Version 2.0).*

Tobiáš, P. (2015, jun). An Investigation into the Possibilities of BIM and GIS Cooperation and Utilization of GIS in the BIM Process. *Geoinformatics FCE CTU*, *14* (1), 65–78. doi: 10.14311/gi.14.1.5

Trometer, S., Schilling, A., Heyer, T. & Mager, K. (2017, April). *Hochwasserrisikoanalyse im urbanen Raum auf der Basis von gekoppelten hydrodynamisch-numerischen Modellen und 3D-Stadtmodellen (Phase 1)* (resreport). CADFEM GmbH, virtualcitySYSTEMS & TU Dresden.

TS 101 903 - V1.3.2 - XML Advanced Electronic Signatures (XAdES) (Bd. 1.3.2; Techical Specification). (2006, März). Zugriff auf https://www.etsi.org/deliver/etsi_ts/101900_101999/101903/01.03.02_60/ts_101903v010302p.pdf

Ueding, M. (2017, August). *The Case for Standard Configuration File Formats.* https://martin-ueding.de/posts/the-case-for-standard-configuration-file-formats/. Zugriff auf https://martin-ueding.de/posts/the-case-for-standard-configuration-file-formats/

Umweltbundesamt. (2018). *Die Stadt für Morgen: Die Vision.* https://www.umweltbundesamt.de/themen/verkehr-laerm/nachhaltige-mobilitaet/die-stadt-fuer-morgen-die-vision. Zugriff auf https://www.umweltbundesamt.de/themen/verkehr-laerm/nachhaltige-mobilitaet/die-stadt-fuer-morgen-die-vision

van den Brink, L., Stoter, J. & Zlatanova, S. (2014). Modeling an application domain extension of CityGML in UML OGC Best Practice. (OGC 12-066). Zugriff auf https://portal.opengeospatial.org/files/?artifact_id=49000

Volksbank am Würtemberg eG. (2016). *Welche Unterlagen werden füreine Baufinanzierung benötigt?* https://www.voba-aw.de/content/dam/f0538-0/Dokumente/Formulare/Checkliste Baufi_ neu.pdf. Zugriff auf https://www.voba-aw.de/content/dam/f0538-0/Dokumente/Formulare/ChecklisteBaufi_neu.pdf

Walsh, N., Milowski, A. & Thompson, H. S. (2010). *XProc: An XML Pipeline Language* (Bericht). W3C. Zugriff auf https://www.w3.org/TR/xproc/

Waser, L. (2020). Wsl berichte oberflächenmodelle aus luftbildern für forstliche anwendungen leitfaden afl 2020. In (Bd. Heft 87, S. 9). Eidg. Forschungsanstalt für Wald, Schnee und Landschaft.

Wasmeier, P. (2018, März). Reduktionen im neuen amtlichen UTM-Lagebezugssystem in der ingenieurgeodätischen Praxis.. https://www.ldbv.bayern.de/file/pdf/12812/InfoVerm2018_Wasmeier.pdf. Zugriff auf https://www.ldbv.bayern.de/file/pdf/12812/InfoVerm2018_Wasmeier.pdf

Web3D Consortium. (o. J.). *Getting started with x3d.* https://www.web3d.org/getting-started-x3d. Zugriff auf https://www.web3d.org/getting-started-x3d

Weise, M., Liebich, T. & Wix, J. (2008, 09). Integrating use case definitions for IFC developments. *ECPPM 2008: EWork and EBusiness in Architecture, Engineering and Construction, ECPPM 2008*, 637–645. doi: 10.1201/9780203883327.ch71

Willenborg, B. (2015). *Simulation of explosions in urban space andresult analysis based on CityGML-City Modelsand a cloud-based 3D-Webclient* (mathesis). Technical University Munich Chair of Geoinformatics.

Wirth, A. & Schneeweiß, A. (2019). Öffentliches baurecht praxisnah. In (S. 9). Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-25720-0

Wätjen, D. (2018). *Kryptographie*. Springer Fachmedien Wiesbaden. doi: 10.1007/978-3-658-22474-5

Yiu, K., Reagle, J., Roessler, T., Eastlake, D., Cantor, S., Datta, P., ... Solo, D. (2015, Juli). *XML Signature Syntax and Processing Version 2.0* (W3C Working Group Note). W3C. Zugriff auf https://www.w3.org/TR/2015/NOTE-xmldsig-core2-20150723/