TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

D I P L O M A R B E I T

Conceptual Modeling and Implementation of a bidirectional data interface between CityGML and EnergyPlus

Ausgeführt am Institut für Geodäsie und Geoinformation
Forschungsgruppe Kartographie
der Technischen Universität Wien

unter der Anleitung von

Univ.Prof. Mag.rer.nat. Dr.rer.nat. Georg Gartner

durch

Holcik Patrick, BSc

Hutweidengasse 19/4/19
1190 Wien
Matrikelnummer: 00509423

Wien, 25. August 2017

Acknowledgements

This work was carried out during an internship at AIT Austrian Institute of Technology, Smart Cities and Regions Research Field, from July 2015 to May 2016 under the supervision of Dr. Giorgio Agugiaro. I am grateful to AIT and especially Dr. Giorgio Agugiaro, a very passionate and knowledgeable scientist for the opportunity to be part of an interesting and challenging field of work. I would like to thank my thesis advisor Dr. Georg Gartner of the Department of Geodesy and Geoinformation Research Group Cartography at the Technical University of Vienna for supporting me with invaluable advice, patience and constructive criticism, whenever necessary. Furthermore, i would like to thank my colleagues Stefan Hauer and Aurelien Bres at the Austrian Institute of Technology, whose experience and knowledge helped me a lot.

Abstract

Virtual 3D City models are increasingly being used for different application areas like urban planning, mobile telecommunication, tourism, disaster management, pedestrian navigation and so on. In recent years most models have been defined purely by their geographical or geometrical aspects, however these models can mostly be used for visualization purposes only. Semantic 3D city models, on the other hand, do not only refer to geometry: all objects included are also described by semantics (e.g. building type, usage, construction date) and topology (e.g. adjacency to other buildings, shared walls), allowing for thematic queries and analysis tasks. With this in mind, the Open Geospatial Consortium (OGC) CityGML standard is designed as an open data model and XML-based format for storage, manipulation, presentation and data exchange of semantic city models at urban and territorial scale. It can further be extended by means of so-called Application Domain Extensions (ADE). This possibility represents a useful characteristic of CityGML: depending on the specific needs, new features or properties can be added. The increasing international acceptance and diffusion of CityGML is proven by the steadily growing number of cities, universities and other research centers, as well as private companies, which are creating and using semantic virtual 3D city models, where CityGML plays more and more an important role as a means to deliver integrated, harmonized and coherent data, which can thus be stored in a centralized, shared database so, that multiple entities of different areas (both public and private) can profit from it. For example, city-wide energy planning requires a precise understanding of the complex system interdependencies at urban level with regards to demand and supply of energy resources, including their spatial distribution. The identification of energetically inefficient buildings therefore plays an important role. Identifying those buildings by modelling their energy performance enables, for example, building professionals to optimize the building design with regard to energy efficiency or at wider scale to define policies at block/district/quartier scale to improve the overall energy demand. Nowadays, a quite large number of specific tools (i.e. software simulation tools) already exist to assess a single building, but their use at urban scale, therefore with multiple buildings and profiting from existing data coming from a semantic 3D city model, is still lacking. Furthermore, the input data, both geometrically and thematically, generally still needs to be prepared manually for each building simulation scenario. The underlying idea of this work is to couple such a simulation tool (namely, Energy Plus) with an already existing CityGML-based 3D city database, in order to exploit the amount of data already available and automatize data ETL (extract transform and load) process, and therefore the whole pipeline enabling the energy simulation of the buildings. The proposed solution includes the conceptual modeling and implementation of an interface able to transfer data bi-directionally between CityGML (and its Energy ADE) and the energy simulation software EnergyPlus. The Energy ADE is being developed by an international consortium of academic and private partners with the goal of extending and enriching CityGML with all those features and attributes that are relevant for energy simulations. EnergyPlus is a whole-building energy simulation program widely used and adopted

internationally to model energy and water use in buildings. EnergyPlus can greatly profit from the amount of integrated information that a CityGML/Energy ADE-based model can offer, ranging from geometries to all other relevant energy-related features and attributes. In order to achieve the proposed solution the following steps are necessary. The CityGML classes will be evaluated and understood, as well as the EnergyPlus Input Data Format. After the evaluation of the existing classes, CityGML classes (and subsequently the Energy ADE as well) need to be mapped into the appropriate EnergyPlus counterpart. In case of missing classes and or missing data, appropriate assumptions need to be made. Afterwards the format of the EnergyPlus simulation results needs to be evaluated, in order to be able to define an interface to translate meaningful results back into CityGML. The last step includes the testing of the bidirectional data interface and the evaluation of the results. Discussed will be under which prerequisites good results can be expected, as well as the pitfalls to be avoided.

Contents

1. Introduction

1.1. Motivation and background

Virtual 3D City models are increasingly being used for different application areas like urban planning, mobile telecommunication, tourism, disaster management, pedestrian navigation and so on. In recent years most models have been defined purely by their geographical or geometrical aspects, however these models can mostly be used for visualization purposes only (see chapter 2 for an overview of existing 3D data models).

Semantic 3D city models, on the other hand, do not only refer to geometry: all objects included are also described by semantics (e.g. building type, usage, construction date) and topology (e.g. adjacency to other buildings, shared walls), allowing for thematic queries and analysis tasks. With this in mind, the Open Geospatial Consortium (OGC) CityGML standard is designed as an open data model and XML-based format for storage, manipulation, presentation and data exchange of semantic city models at urban and territorial scale. It can further be extended by means of so-called Application Domain Extensions (ADE). This possibility represents a useful characteristic of CityGML: depending on the specific needs, new features or properties can be added.

The increasing international acceptance and diffusion of CityGML is proven by the steadily growing number of cities, universities and other research centers, as well as private companies, which are creating and using semantic virtual 3D city models, where CityGML plays more and more an important role as a means to deliver integrated, harmonized and coherent data, which can thus be stored in a centralized, shared database that multiple entities of different areas (both public and private) can profit from it eventually.

For example, city-wide energy planning requires a precise understanding of the complex system interdependencies at urban level with regards to demand and supply of energy resources, including their spatial distribution. According to European Directive 2010/31 buildings account for around 40% of total energy consumption in the European Union. With this sector expanding, one may reckon with an increase in energy consumption. The goal is to drastically reduce energy consumption and to propagate renewable energy resources [Union, 2010]. The identification of energetically inefficient buildings therefore plays an important role. Identifying those buildings by modeling their energy performance enables, for example, building professionals to optimize the building design with regard to energy efficiency or at

wider scale to define policies at block/district/quarter scale to improve the overall energy demand.

Quite a large number of specific tools (i.e. software simulation tools) already exist to assess a single building providing detailed energy performance data with a fine-grained temporal resolution, but their use at urban scale, therefore with multiple buildings and profiting from existing data coming from a semantic 3D city model, is still lacking. Furthermore, the input data, both geometrically and thematically, generally still needs to be prepared manually for each building simulation scenario in a time-consuming manner [Kaden and Kolbe, 2013, Agugiaro, 2015]. Nowadays city-wide energy performance calculation often involve top-down statistical approaches based on econometric and technological data or bottom-up approaches based on statistical and engineering data resulting in often coarse spatial results and time resolution (yearly or monthly estimations per building) also due to heterogeneous and incomplete or missing data [Kaden and Kolbe, 2013, Caputo et al., 2013]. The underlying idea is to exploit data available from an existing integrated 3D city model and to couple it with a dynamic simulation software, in order to enjoy the benefits of a fully-fledged dynamic simulation tool (e.g. enhanced time resolution from monthly to hourly values and the possibility to use real weather data). Therefore, the question to answer is, whether it is possible to link a standardized, spatio-semantic coherent 3D city model (CityGML) into already existing dynamic software simulation tools (EnergyPlus) to create precise hourly energy performance data at urban scale on building level to provide planning basis for city-wide energy-planning. To be more general, the broader question is to evaluate, whether a standardized, spatio-semantic coherent 3D city model can be used as basis for different kind of applications. Data thus obtained furthermore needs to be made available and visualized in a manner so that multiple entities of different areas can further profit from it. According to [Altmaier and Kolbe, 2003] 3D visualization not only provides graphical representations, but furthermore contributes to the improvement of work flows and efficiency in multiple application fields by supporting analysis, decision making, management and planning. Integrating the results into a 3D WebGIS would allow for easy access to geospatial data via Web services.

In [Banfi, 2013] some groundwork was done, as a way to create a UML model of the EnergyPlus Input Data Dictionary (IDD) was implemented. Furthermore a mapping between CityGML LoD2 and LoD3 geometries and their EnergyPlus equivalents was realized, though with some constraints. The proposed solution considered thematic surfaces and no other properties and was also limited to 1 building with 1 thermal zone.

[Agugiaro, 2016b] and [Agugiaro and Möller] describe the selection, analysis, preparation, integration of a number of different heterogenous data sets with the goal of creating a semantic virtual city model of Vienna. The work described is part of the European Marie-Curie ITN project "*CINERGY, Smart cities with sustainable energy systems.*" aiming at developing urban decision making and operational optimisation software tools in order to minimise non-renewable energy use in cities

[url, 2016a]. Some results can be seen at the "*3D city model of Vienna-Meidling*[1]" Geographic Information System (GIS) (see figure 1.1) developed at the Austrian Institute of Technology[2], which provides energy related information on a number of buildings in the 12th district of Vienna/Austria on a yearly basis. Automated dynamic energy performance simulations could greatly enhance data quality with finer-grained sub-hourly temporal resolution.

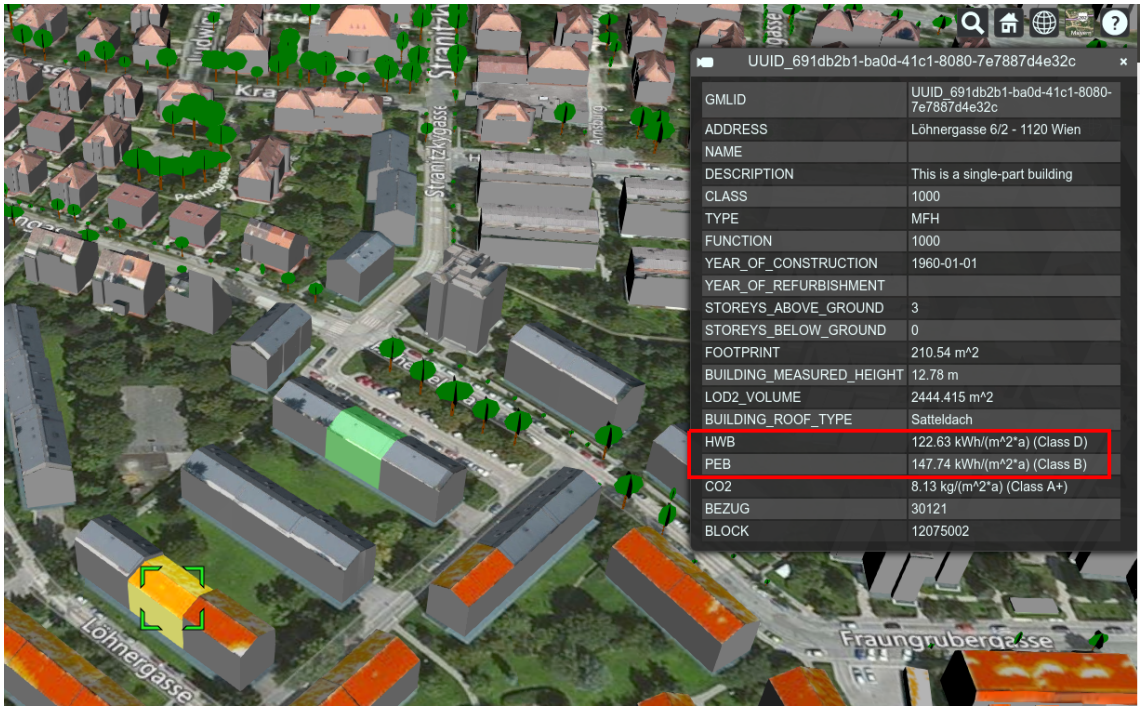

Figure 1.1.: 3D city model of Meidling in Vienna/Austria with yearly energy related data.

1.2. Proposed solution

The underlying idea of this work is to couple such a simulation tool (namely, EnergyPlus) with an already existing CityGML-based 3D city database, in order to exploit the amount of data already available and automatize data ETL (extract transform and load) process, and therefore the whole pipeline enabling the energy simulation of the buildings. The proposed solution includes the conceptual modeling and implementation of an interface able to transfer data bi-directionally between CityGML (and its Energy ADE) and the energy simulation software EnergyPlus. The Energy ADE is being developed by an international consortium of academic and private partners with the goal of extending and enriching CityGML with all those features and attributes that are relevant for energy simulations. EnergyPlus is a dynamic whole-building energy simulation program widely used and adopted internationally

[1]http://sbc1.ait.ac.at:10180/projects/meidling/cesium/webmap/index.html
[2]http://www.ait.ac.at/

to model energy and water use in buildings. EnergyPlus can greatly profit from the amount of integrated information that a CityGML/Energy ADE-based model can offer, ranging from geometries to all other relevant energy-related features and attributes. In order to achieve the proposed solution the following steps are necessary. The CityGML classes will be evaluated and understood, as well as the EnergyPlus Input Data Format. After the evaluation of the existing classes, CityGML classes (and subsequently the Energy ADE as well) need to be mapped into the appropriate EnergyPlus counterpart. In case of missing classes and or missing data, appropriate assumptions need to be made. Afterwards the format of the EnergyPlus simulation results needs to be evaluated, in order to be able to define an interface to translate meaningful results back into CityGML. The last step includes the testing of the bidirectional data interface and the evaluation of the results, as well as providing a meaningful way to communicate the simulation results in a convenient and comprehensive way. Discussed will be under which prerequisites good results can be expected, as well as the pitfalls to be avoided.

2. Overview of 3D Data Models

2.1. Introduction

This chapter gives a short overview of existing 3D data models that are being utilized in different domains serving different areas of application with a focus an city models and their ability to represent energy related information. In general it can be said, that 3D data models are of 2 types namely *geometrical models* and *semantic models*. *Geometrical 3D data models* define entities purely by the geometrical representation using different spatial objects (like points, line strings, polygons, and so on). *Semantic 3D data models* on the other hand add non-spatial characteristics and relationships amongst the entities.

2.2. COLLADA

COLLADA[1] is an XML-based, open standard data model used as an interchange file format for interactive 3D applications (like ArcGIS, Google Earth or SketchUp). It is developed by the Khronos Group[2] and has been adopted as an ISO standard in 2012. The model focuses on geometric representation using primitives like *lines*, *linestrips*, *polylist*, *polygons*, *triangles*, *trifans* and *tristrips*. It furthermore supports *topologies*, *physics*, *animation*, *kinematics* and also provides a mechanism of *multi-versioning* the same object. The model does not support semantics and has no mechanism to model different Level of Details (LOD) [Barnes and Finch, 2008, Kumar et al., 2016, url, 2016b].

2.3. gbXML

GbXML[3] is an XML-based open data model, that allows to transfer between disparate 3D building information models (BIM) architectural/engineering analysis software tools. It is supported and adopted by leading BIM vendors (like Autodesk, Trimble, Graphisoft and Bentley), due to its extensive coverage of building energy domains, becoming the de facto industry standard. GbXML was launched in 2000, becoming the draft schema for the Building Performance & Analysis Working

[1] `https://www.khronos.org/collada/`
[2] `https://www.khronos.org/`
[3] `http://www.gbxml.org/About_GreenBuildingXML_gbXML`

Group. Being a standalone entity since 2009 gbXML is funded by organizations like the U.S. Department of Energy and the National Renewable Energy Lab (NREL) amongst others [url, 2016e, Kumar et al., 2016]. The model allows for storing geometrical representation, thematic differentiation (like *InteriorWall*, *ExteriorWall*, *Roof*, *Floor* and so on) and even supports storing energy related data (like space compositioning trough thermal zones, building construction information, internal loads and operation schedules) [Kumar et al., 2016, Moon et al., 2011].

2.4. IFC

IFC[4] is an open object oriented BIM file format and data model developed by buildingSMART International[5] (formerly known as the International Alliance for Interoperability IAI) to facilitate interoperability between applications in the architecture, engineering and construction industry. IFC data files are exchanged between applications using the following formats: *STEP*, *XML*, *PKzip*. IFC supports multiple geometrical models, which are independent from the semantic implementation. An IFC building (*IfcBuilding*) can have multiple storeys (*IfcStorey*) with multiple rooms (*IfcSpace*). Building elements can have multiple openings like doors and windows (*IfcDoor* and *IfcWindow*, respectively). IFC does not support a mechanism to depict multiple Level of Detail (LOD) [url, 2016g, Kumar et al., 2016].

2.5. KML

KML[6] is an XML-based open data model used to encode and transport geographical representations for 2D or 3D visualization purposes [Wilson, 2008]. It was originally created by Keyhole, Inc. and was submitted to the Open Geospatial Consortium (OGC) by Google in 2007 and subsequently got adopted as an OpenGIS standard in 2008 [Sandvik, 2008]. KML supports primitive geometries like points (*kml:Point*), line strings (*kml:LineString*), linear rings (*kml:LinearRing*), polygons (*kml:Polygon*) and multi geometries (*kml:MultiGeometry*). KML does not support semantics and has no mechanism to support different Level of Detail (LOD) for buildings [Wilson, 2008].

2.6. INSPIRE

The INSPIRE[7] (Infrastructure for Spatial Information in the European Community) directive (2007/2/EC) aims at establishing an european-wide Spatial Data Infrastructure (SDI) for the purpose of EU environmental policies, as spatial information

[4] `http://www.buildingsmart-tech.org/specifications/ifc-overview`
[5] `http://www.buildingsmart-tech.org/`
[6] `http://www.opengeospatial.org/standards/kml`
[7] `http://inspire.ec.europa.eu`

in Europe can be described as heterogeneous with fragmented datasets of different sources, with gaps in availability, lacking in harmonization and interoperability between datasets at different geographical scales[INSPIRE, 2013, Bartha and Kocsis, 2011]. INSPIRE consists of five components: *Interoperability of Spatial Data Sets and Services*, enabling the possibility to combine spatial data and services from different sources in a consistent way , *Metadata* to allow the discovery and evaluation of data sets and services, *Network Services* to make it possible to discover, transform, view, download and process spatial data, *Data Sharing* for easy data exchange between public bodies and to allow third parties to have free and easy access, as well as *Coordination and Complementary Measures* to monitor the organizational and management aspects of the INSPIRE implementation[INSPIRE, 2013, Bartha and Kocsis, 2011].

The INSPIRE directive defines 34 spatial data themes covering a wide range from *Addresses*, *Administrative Units*, *Cadastral Parcels*, *Energy Resources*, *Geology*, *Landcover*, *Natural Resources* amongst others (see `https://inspire.ec.europa.eu/Themes/126/2892`). Of interest is the data theme *Buildings*, which will be discussed here.

The INSPIRE *Buildings* data theme consists of 3 applications schemas (which are conceptual models related to specific areas) namely *BuildingsBase*, *Buildings2D* and *Buildings3D* and 3 additional application schemas namely *BuildingsExtendedBase*, *BuildingsExtended2D* and *BuildingsExtended3D*[INSPIRE, 2013]. See figure 2.1.

BuildingsBase describes the concepts common to all following application schemas, combining the core normative semantics. *Buildings2D* and *Buildings3D* describe the 2D and 3D geometric representation of the buildings, respectively. *BuildingsExtendedBase* describes additional semantics of the *BuildingsBase* profile. *BuildingsExtended2D* and *BuildingsExtended3D* profiles serve as an extension to *Buildings2D* and *Buildings3D*, respectively, to include additional spatial object types for example installations and other constructions or to provide additional concepts to be able model more detailed information about buildings and their associated objects (for example walls, roofs, textures)[INSPIRE, 2013].

The INSPIRE *Buildings* model is based on the *CityGML* standard (discussed in detail in chapter 3), although a series of changes have been introduced like adding common INSPIRE theme attributes to ensure cross-theme consistency, modifying existing CityGML attributes and most notably simplifying the geometrical and appearance model of CityGML[INSPIRE, 2013].

The INSPIRE *Buildings* model provides some very basic energy-related attributes considering the building itself (for example connectionToElectricity,heatingSource, heatingSystem, energyPerformance, materials, etc.) or to wall and roof surfaces (materialOfRoof/Facade).

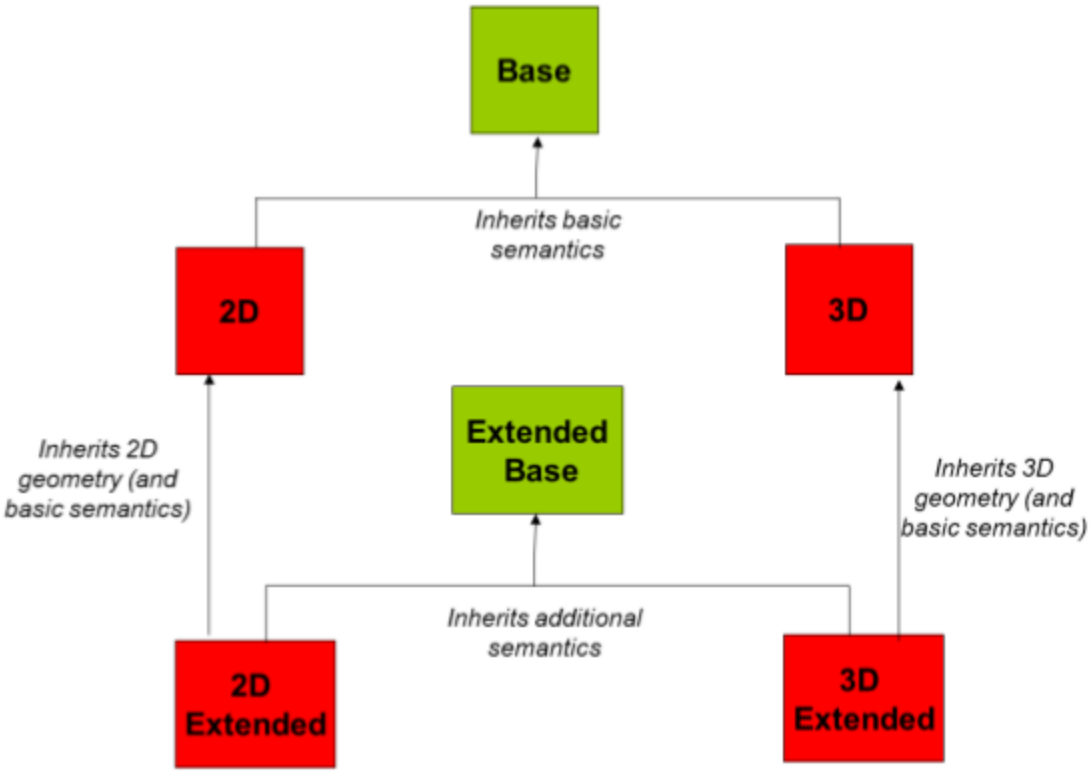

Figure 2.1.: INSPIRE application schemas. Source: INSPIRE Data Specification on Buildings Technical Guidelines[INSPIRE, 2013].

2.7. Summary

A lot of 3D data models have been developed for the purpose of modeling, visualization, storage and exchange of data, which ultimately differ regarding geometry, semantics, Level of Detail (LOD) and schema[Kumar et al., 2016], as outlined in section 2.2 - 2.5. Of the discussed 3D data models some provide very detailed geometrical and semantical information but are limited to single buildings (gbXML, IFC), often very time consuming to create and with limited availability [Agugiaro, 2015]. Others provide purely geometric representations, omitting semantical information, which narrows their use of application (COLLADA, KML). In order to overcome this heterogeneity, an integrated semantically data model combining detailed geometrical, topological and semantical information is needed.

The INSPIRE Buildings model provides these features, but lacks in term of providing energy-related attributes. To these extents, CityGML with its extension mechanism called ADE (discussed in detail in chapter 3 and 4) is conceived to solve exactly these problems [Agugiaro, 2015]. The CityGML schema can simply be extended to provide all the necessary energy-related attributes needed. Figure 2.2 shows CityGML as the missing link between detailed single building models and the more general purpose INSPIRE building model. The following chapter discusses CityGML in more detail.

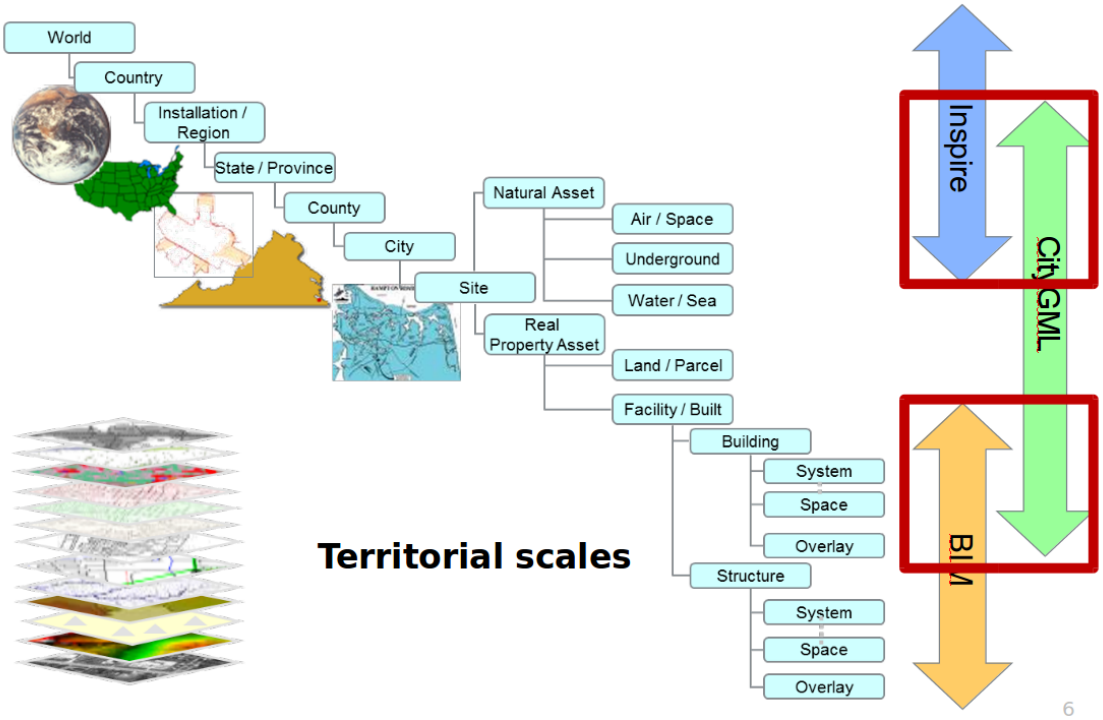

Figure 2.2.: Territorial Scales of data models. Source: Austrian Institute of Technology.

3. CityGML

3.1. Introduction

This chapter contains a general introduction to the CityGML standard. At first a short overview about the motivation behind the implementation as well as the development history is given, followed by a description of the most significant features in more detail. Since CityGML covers a wide variety of possible applications, this chapter is restricted to the features most significant for this master thesis.

3.2. General

3D city models and their applications are gaining more and more importance. Examples are noise propagation, urban and telecommunication planning, disaster management or noise propagation simulation and mapping. Simultaneously municipalities, local or national government surveying agencies, commercial and other organizations are increasingly building and maintaining 3D city models. For most of these applications only the geometrical and graphical aspects may be insufficient. Depending on the application, information on the semantic level is of interest. Typically providers and users of 3D city models are different. Often data needed is gathered from different sources, which were collected at different times with different levels of detail, which is why users face the problem of data heterogeneity [Gröger and Plümer, 2012]. CityGML tries to solve these problems (see figure 3.1).

CityGML[1] is an XML-based, open data model used for the storage, manipulation, presentation and data exchange of semantic virtual 3D city models at urban and territorial scale [Gröger and Plümer, 2012]. CityGML defines a generic model describing geometry, topology, semantics and appearance of 3D objects in urban environments, with five possible levels of (geometric and semantic) detail (LoD). Included are also generalization hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. All city objects can be connected by external links to specific ancillary data repositories (e.g. cadastral data, solar yield estimation of the roofs, etc.) For specific domain areas, CityGML also provides an extension mechanism to enrich the data with identifiable features under preservation of semantic interoperability by means of so-called Application Domain Extensions (ADE).

[1] http://www.citygml.org/

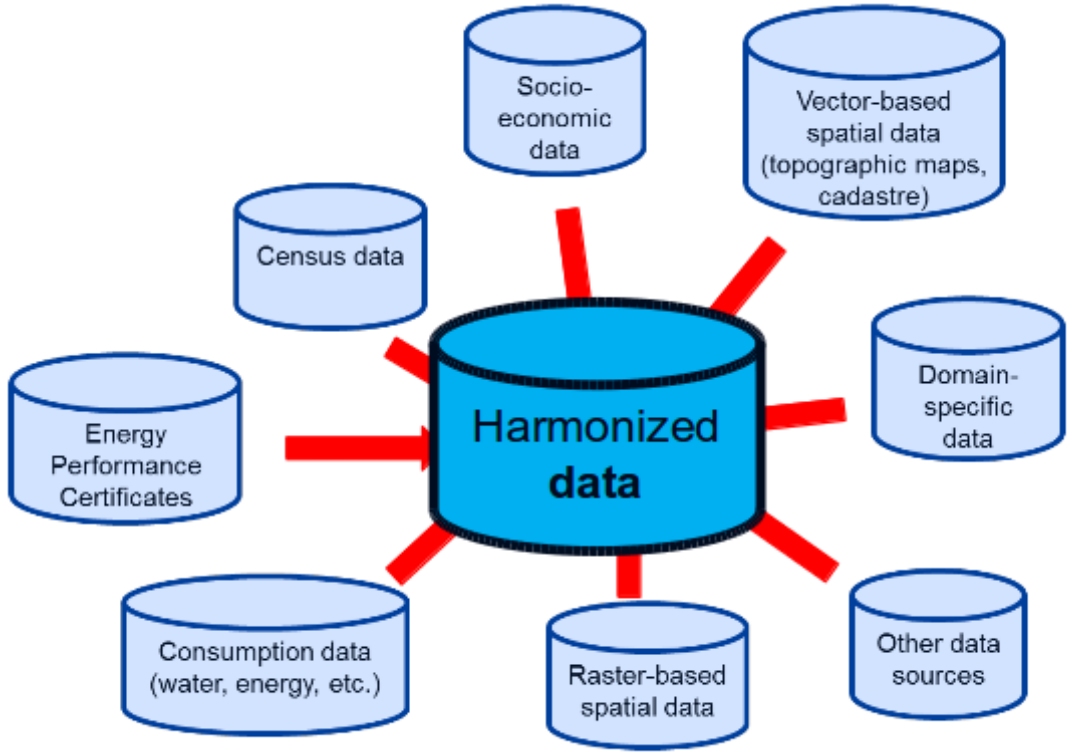

Figure 3.1.: Harmonizing different data sources. Source: Austrian Institute of Technology.

CityGML is implemented as an application schema of the Geography Markup Language 3.1.1 (GML3) (see [Portele, 2007]), which is the international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium[2] (OGC) and the ISO TC211 [Gröger et al., 2012]. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS [Gröger et al., 2012].

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D) and since 2010 as part of the Spatial Data Infrastructure Germany (GDI-DE) initiative. Adopted in 2008 as an Open Geospatial Consortium (OGC) standard at version 1.0 and updated as a majors revision to version 2.0 in 2012, which introduces "substantial additions and new features to the thematic model of CityGML", as well as retaining backwards compatibility[Gröger et al., 2012].

An example of the CityGML format can be seen in listing 3.1.

[2]http://www.opengeospatial.org/

```xml
<?xml version="1.0" encoding="utf-8"?>
<core:CityModel xsi:schemaLocation="http://www.opengis.net/citygml/1.0 http://
    schemas.opengis.net/citygml/1.0/cityGMLBase.xsd http://www.opengis.net/
    citygml/building/1.0 http://schemas.opengis.net/citygml/building/1.0/building
    .xsd" xmlns:core="http://www.opengis.net/citygml/1.0" xmlns="http://www.
    opengis.net/citygml/profiles/base/1.0" xmlns:bldg="http://www.opengis.net/
    citygml/building/1.0" xmlns:grp="http://www.opengis.net/citygml/
    cityobjectgroup/1.0" xmlns:gml="http://www.opengis.net/gml" xmlns:xAL="urn:
    oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:xlink="http://www.w3.org/1999/
    xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<gml:name>AC14-FZK-Haus</gml:name>
<gml:boundedBy>
  <gml:Envelope srsDimension="3" srsName="urn:adv:crs:ETRS89_UTM32">
    <gml:lowerCorner srsDimension="3">458877 5438353 -0.2 </gml:lowerCorner>
    <gml:upperCorner srsDimension="3">458889 5438363 6.317669 </gml:upperCorner>
  </gml:Envelope>
</gml:boundedBy>
<core:cityObjectMember>
  <bldg:Building gml:id="UUID_d281adfc-4901-0f52-540b-4cc1a9325f82">
    <gml:description>FZK-Haus</gml:description>
    <gml:name>AC14-FZK-Haus</gml:name>
    <core:creationDate>2010-12-01</core:creationDate>
    <bldg:class>1000</bldg:class>
    <bldg:function>1000</bldg:function>
    <bldg:usage>1000</bldg:usage>
    <bldg:yearOfConstruction>2020</bldg:yearOfConstruction>
    <bldg:roofType>1030</bldg:roofType>
    <bldg:measuredHeight uom="m">6.52</bldg:measuredHeight>
    <bldg:storeysAboveGround>2</bldg:storeysAboveGround>
    <bldg:storeysBelowGround>0</bldg:storeysBelowGround>
    <bldg:boundedBy>
      <bldg:WallSurface gml:id="GML_5856d7ad-5e34-498a-817b-9544bfbb1475">
      <gml:name>Outer Wall 1 (West)</gml:name>
      <bldg:lod2MultiSurface>
      <gml:MultiSurface>
        <gml:surfaceMember>
          <gml:Polygon gml:id="PolyID7350_878_759628_120742">
            <gml:exterior>
              <gml:LinearRing gml:id="PolyID7350_878_759628_120742_0">
                <gml:pos>458877 5438358 6.31769145362398 </gml:pos>
                <gml:pos>458877 5438363 3.43094010767585 </gml:pos>
                <gml:pos>458877 5438363 -0.2 </gml:pos>
                <gml:pos>458877 5438353 -0.2 </gml:pos>
                <gml:pos>458877 5438353 3.43094010767585 </gml:pos>
                <gml:pos>458877 5438358 6.31769145362398 </gml:pos>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </gml:surfaceMember>
      </gml:MultiSurface>
      </bldg:lod2MultiSurface>
      </bldg:WallSurface>
    </bldg:boundedBy>
```

Listing 3.1: CityGML syntax example.

3.3. Main characteristics

In this section an overview of CityGML will be given. The focus lies on the most important aspects: modularisation, multi-scale modelling, the representation of geometry and topology, the ability to reference external data sets and the mechanism to extend CityGML to enrich the data by means of so-called Application Domain Extensions (ADE).

3.3.1. Modularisation

The CityGML data model consists of class definitions for the most important types of objects that one could encounter in virtual 3D city models. These classes have been labelled as to be either mandatory or important in many different areas of applications. Implementations however, are not required to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. Hence, modularisation is applied to the CityGML data model [Gröger et al., 2012].

The CityGML data model is thematically divided into a core module and thematic extension modules. The core module "defines the basic components of the CityGML data model. Thematic extension modules, which are based on the core module, are used to cover specific thematic fields of virtual 3D city models. In CityGML these thematic extension modules are: Appearance, Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody, and TexturedSurface. Modules and their schema dependencies can be seen in figure 3.2 [Gröger et al., 2012].

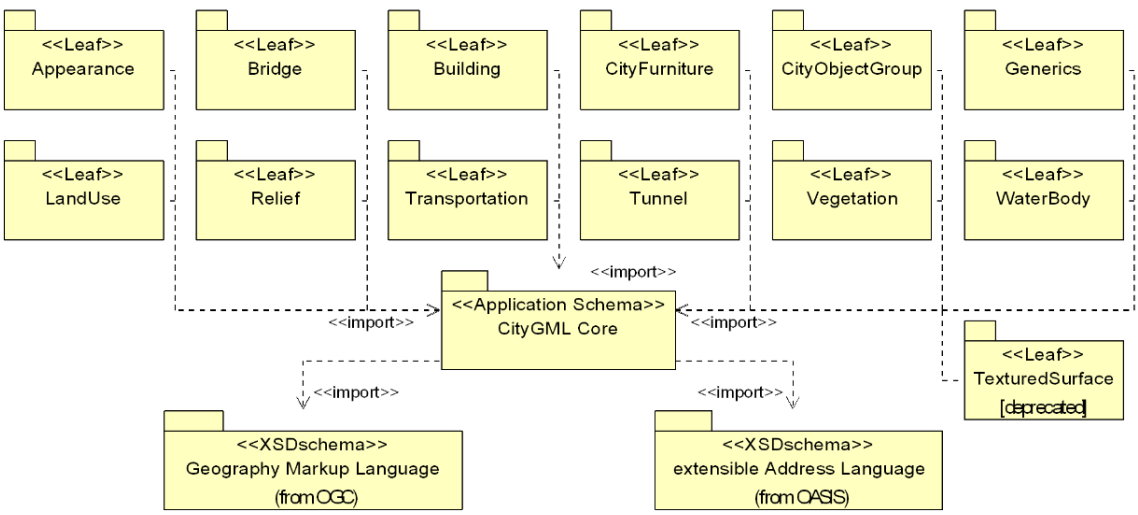

Figure 3.2.: UML package diagram describing the modules of CityGML. Source: [Gröger et al., 2012]

CityGML-compliant implementations may support any combination of extension modules in combination with the core module. These combinations of modules are called "CityGML profiles" [Gröger et al., 2012].

The core module and the thematic building module will be further discussed in detail in section 3.5.

3.3.2. Multi-scale modelling

CityGML presents a multi-scale model with 5 well-defined consecutive Levels of Detail (LOD), where objects are more detailed with increasing LOD regarding both their geometry and thematic differentiation. A CityGML file can contain multiple representations and geometries for each object in different LOD simultaneously [Kolbe, 2009]. The LoD concept also covers semantic aspects. It is not limited to geometrical ones [Gröger and Plümer, 2012]. Figure 3.3 provides an overview of the different levels of detail.

	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	very high
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*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*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, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
…to be continued for the other feature themes					

Figure 3.3.: LOD 0-4 of CityGML with their proposed accuracy requirements. Source: [Gröger et al., 2012]

LOD0 basically represents a 2.5D Digital Terrain Model (DTM). LOD1 is the blocks model without any roof structures. LOD2 can represent roof structures and larger building installations as well as thematically differentiated boundary surfaces. LOD3 realizes detailed wall and roof structures, windows and openings. LOD4 completes LOD3 by adding interior structures for buildings [Gröger et al., 2012, Kolbe, 2009].

A visual example of the Levels of Detail can be seen in figure 3.4.

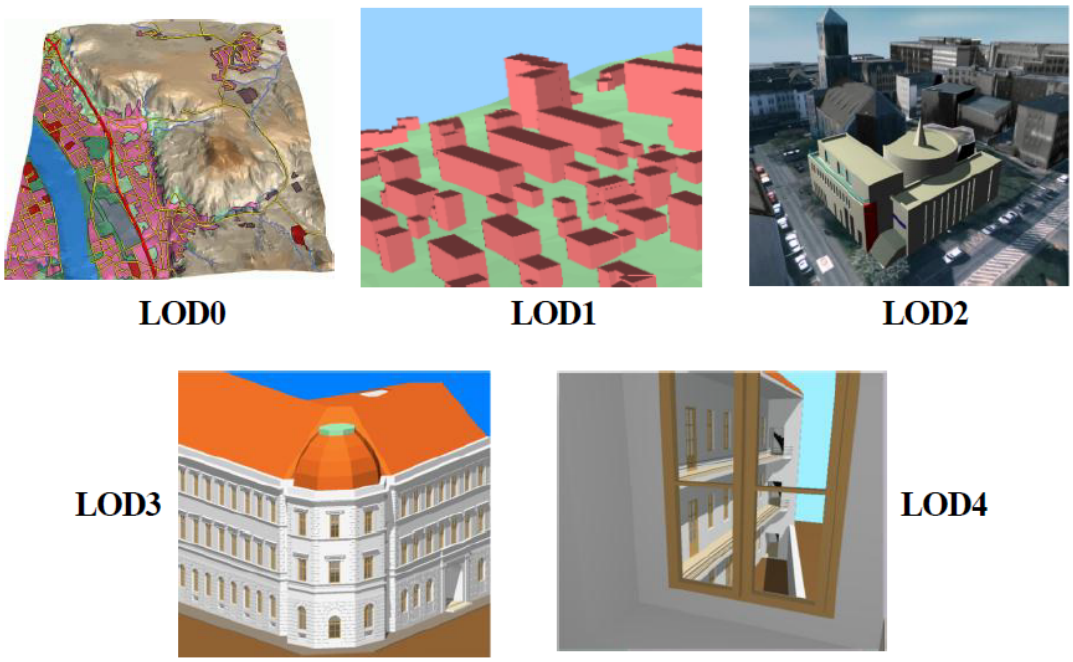

Figure 3.4.: The five levels of detail (LOD) defined by CityGML. Source: [Gröger et al., 2012]

3.3.3. Coherent semantical-geometrical modelling

Unlike conventional approaches of representing 3D city models, where the main focus lied purely on graphical representation, CityGML extends this approach with associated semantic information in order to describe urban models not only in their appearance, but also semantically and topologically (see figure 3.5) [Gröger and Plümer, 2012].

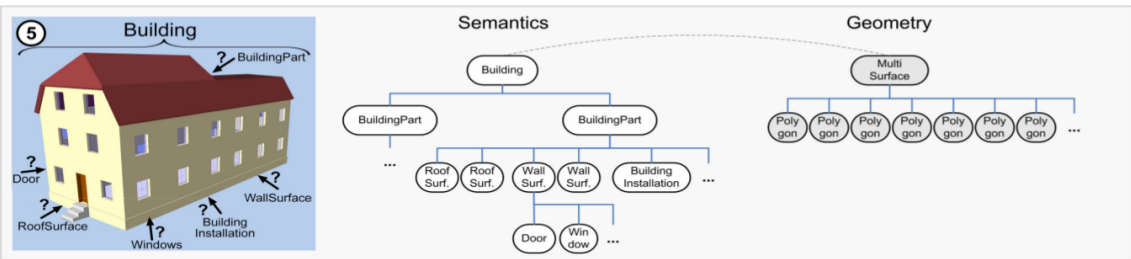

Figure 3.5.: Complex object with fully coherent spatio-semantic structure. Source: [Stadler and Kolbe, 2007]

At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms described by different attributes with relations and aggregation hierarchies (part-whole-relations) between them. At the spatial level, geometry objects are assigned to features representing their location and extent. The separation into 2 hierarchies (semantic and geometrical) in which the objects

are linked together brings a big advantage into play: both hierarchies can be navigated in arbitrarily and even between them. Though it must be noted, that if both hierarchies exist it must be ensured that they fit together. For example a semantically defined window must be geometrically defined as well. [Gröger et al., 2012].

3.3.4. External references

In CityGML, all 3D objects can have relations to external objects in other databases or datasets. The reference of a 3D object to its corresponding object in an external data set is essential, for example if additional data is required, e.g. the name and address of a buildings owner in a cadastral information system (Figure 5). In order to supply such information, each _CityObject may refer to external data sets using the concept of ExternalReference. Such a reference denotes the external information system and the unique identifier of the object in this system. The generic concept of external references allows for any _CityObject an arbitrary number of links to corresponding objects in external information systems.

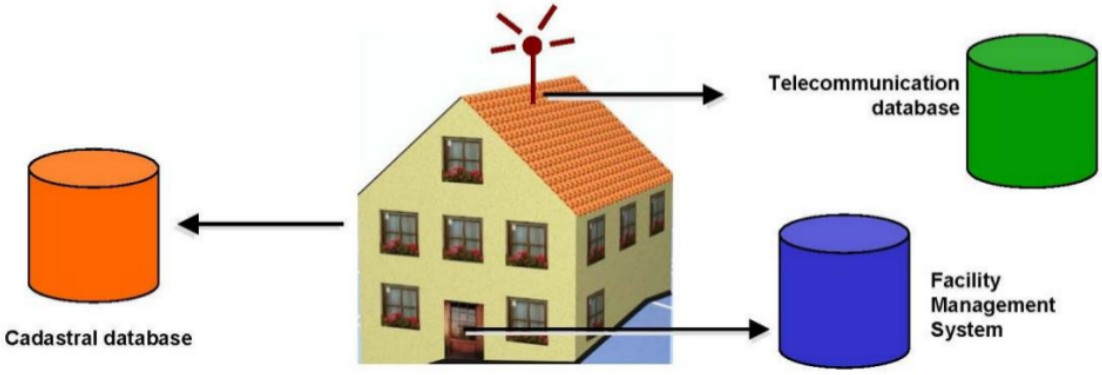

Figure 3.6.: External references. Source: [Gröger et al., 2012]

3.3.5. Extendibility

Though CityGML is designed as a broad universal information model, different applications may be in need of attributes not explicitly modelled in CityGML. Moreover, there may as well the need for objects not covered by thematic classes of CityGML. To counteract, CityGML provides 2 concepts to exchange data: 1) generic objects and attributes, b) Application Domain Extensions (ADE) [Gröger et al., 2012].

Generic objects and attributes allows for the extension of CityGML applications during runtime, meaning any _CityObject may be enriched by additional attributes without any change of the CityGML XML schema. Additionally, features not covered by predefined thematic classes may be modelled using generic objects [Gröger et al., 2012].

Application Domain Extensions (ADE) on the other hand specify additions to the CityGML data model. Such additions introduce new properties to existing CityGML classes or the definition of new object types. ADEs, unlike generic objects and attributes, need to be defined in an extra XML schema definition file making the extension formally specified. ADEs provide a high flexibility in supplying additional information to the CityGML data model. Several ADEs for different application areas are being developed and maintained by information communities. The EnergyADE as an example will be discussed in detail in section 3.3.5.

3.4. Spatial Model

As mentioned earlier, CityGML includes both spatial and thematic models (see section 3.5).

The spatial model allows the consistent and homogeneous definition of geometrical and topological properties of CityGML features, represented by objects of GML3's geometry model, which is based on the ISO 19107 Spatial Schema (see [Herring, 2001]). CityGML uses only a subset of the GML3 geometry as seen in fig. 3.7. GML3 provides a geometrical primitive for each dimension: a 0D object is a Point, a 1D object is a a _Curve, a 2D object is a _Surface, and a 3D object is a _Solid [Gröger et al., 2012].

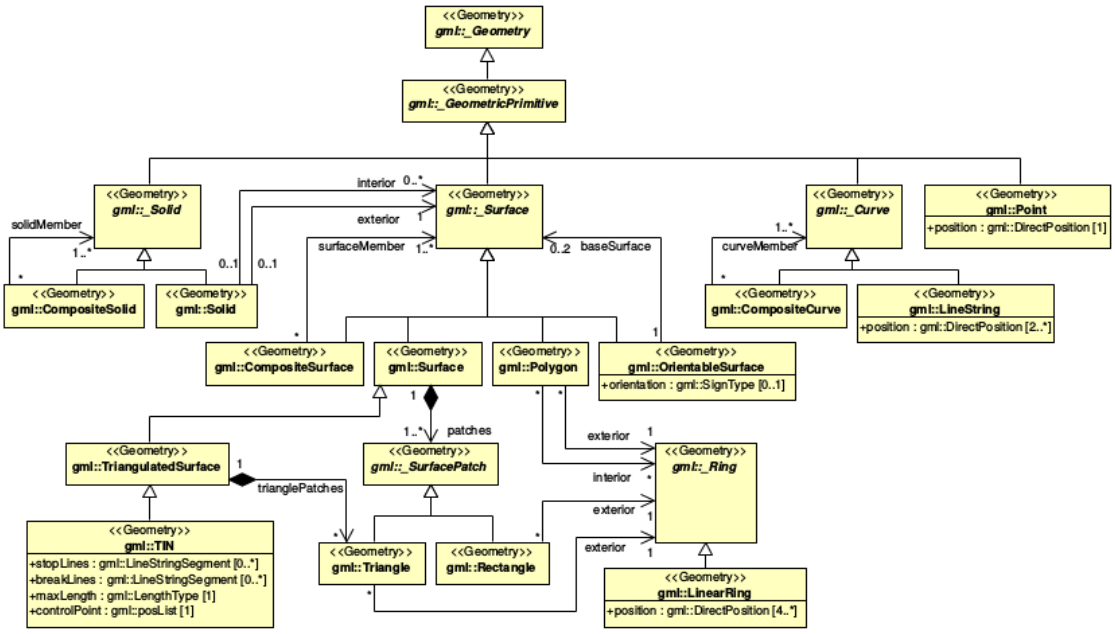

Figure 3.7.: CityGML geometry model (subset of GML3). Source: [Gröger et al., 2012]

The primitives depicted in 3.7 can be combined to form complexes, composites or aggregates (see fig. 3.8). Of importance are the objects Solid and MultiSurface which will be further discussed in section 3.5.2.

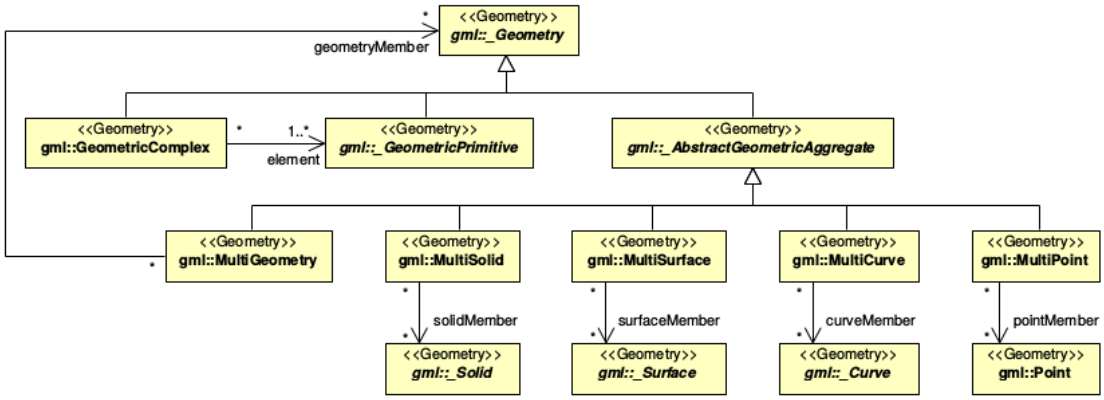

Figure 3.8.: CityGML geometry model (subset of GML3) - complexes and aggregates. Source: [Gröger et al., 2012]

CityGML provides a simple way of representing topology using the XML concept of links (XLink). Each geometry object that should be shared either by geometric aggregates or thematic features can be assigned a unique identifier, which may be referenced using a href attribute simplifying topological representation [Gröger et al., 2012].

A further definition of the spatial model can be found in the CityGML specification document [Gröger et al., 2012] section 8.

3.5. Thematic Model

As already mentioned in section 3.3.1 the CityGML data model is thematically divided into a core module and thematic extension modules (see fig. 3.2).The basic concepts and components of the CityGML belong to the core module. Extensions are used to cover specific thematic fields of virtual 3D city models. In section 3.5.1 and section 3.5.2 the two most important modules, namely core and building, will be discussed in detail.

3.5.1. Core

The CityGML core module, which is the most important, as it contains the basic components and concepts of the data model (see fig 3.9). Every thematic extension needs to implement this module.

The abstract class _CityObject serves as the base class of all thematic classes within the data model. It inherits the attributes gml:id, name, description from GML3's abstract class _Feature. It furthermore provides a creationDate and terminationDate to allow histories of features, as well as qualitative attributes relativeToTerrain and relativeToWater. External References as discussed in section 3.3.4 can be modelled via ExternalReference. The property generalizesTo provides a mechanism to relate a

feature to its corresponding generalized counterpart (in a different LOD). Subclasses of the _CityObject may be aggregated to a single CityModel (which is itself a GML3 FeatureCollection) [Gröger et al., 2012].

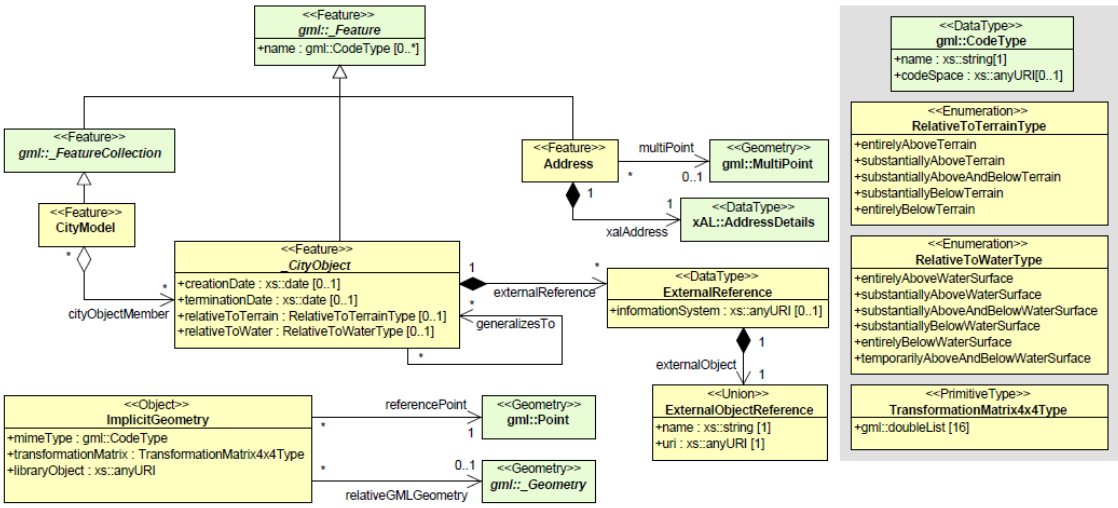

Figure 3.9.: CityGML Core Module. Source: [Gröger et al., 2012]

3.5.2. Building

The building model is the main thematic model this master thesis focuses on. It is one of the most detailed thematic concepts of CityGML. Figure 3.10 shows the UML diagram of the building module. The key class of the model is _AbstractBuilding, which is a subclass of the thematic class _Site (and transitively _CityObject). _AbstractBuilding can be specialised to either a Building or a BuildingPart. Since an _AbstractBuilding consists of BuildingParts, which again are "_AbstractBuildings", an aggregation hierarchy of arbitrary depth may be realised (see fig. 3.11) [Gröger et al., 2012].

Building and BuildingPart inherit the attributes of _AbstractBuilding: the class of the building, the function (e.g. residential, public, or industry), the usage, the year of construction, the year of demolition, the roof type, the measured height, and the number and individual heights of the storeys above and below ground. Address features can be assigned to Buildings or BuildingParts as well [Gröger et al., 2012].

BuildingInstallation is used for modelling smaller features of the building, generally named outer building installations. Typical candidates for this class are chimneys, dormers, balconies, outer stairs, or antennas. BuildingInstallations may only be included from LoD2 models [Gröger et al., 2012].

In LOD3, openings in a building like windows and doors may be modelled by the abstract class _Opening and the derived subclasses Window and Door. With respect to the exterior building shell, the LOD4 data model is identical to that of LOD3. LOD4, which is the highest level of resolution, however also provides the possibility

to model the interior structure of a building with the classes Room and IntBuildingInstallation (see fig. 3.11 for an example of different LODs of a building model) [Gröger et al., 2012].

Interior installations of a building, i.e. objects within a building which (in contrast to furniture) cannot be moved, are represented by the class IntBuildingInstallation. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the Room class, otherwise (e.g. in case of rafters or pipes) with _AbstractBuilding. When it comes to geometry, there are several possibilities to represent a building, depending on the LOD and the type of geometry, either as solid or as boundary surface. Of particular relevance is here that buildings can be represented from LOD2 also by means of thematic surfaces, in that, for example, geometries of the outer shell are classified in to OuterWall, GroundSurface, RoofSurface, etc [Gröger et al., 2012].

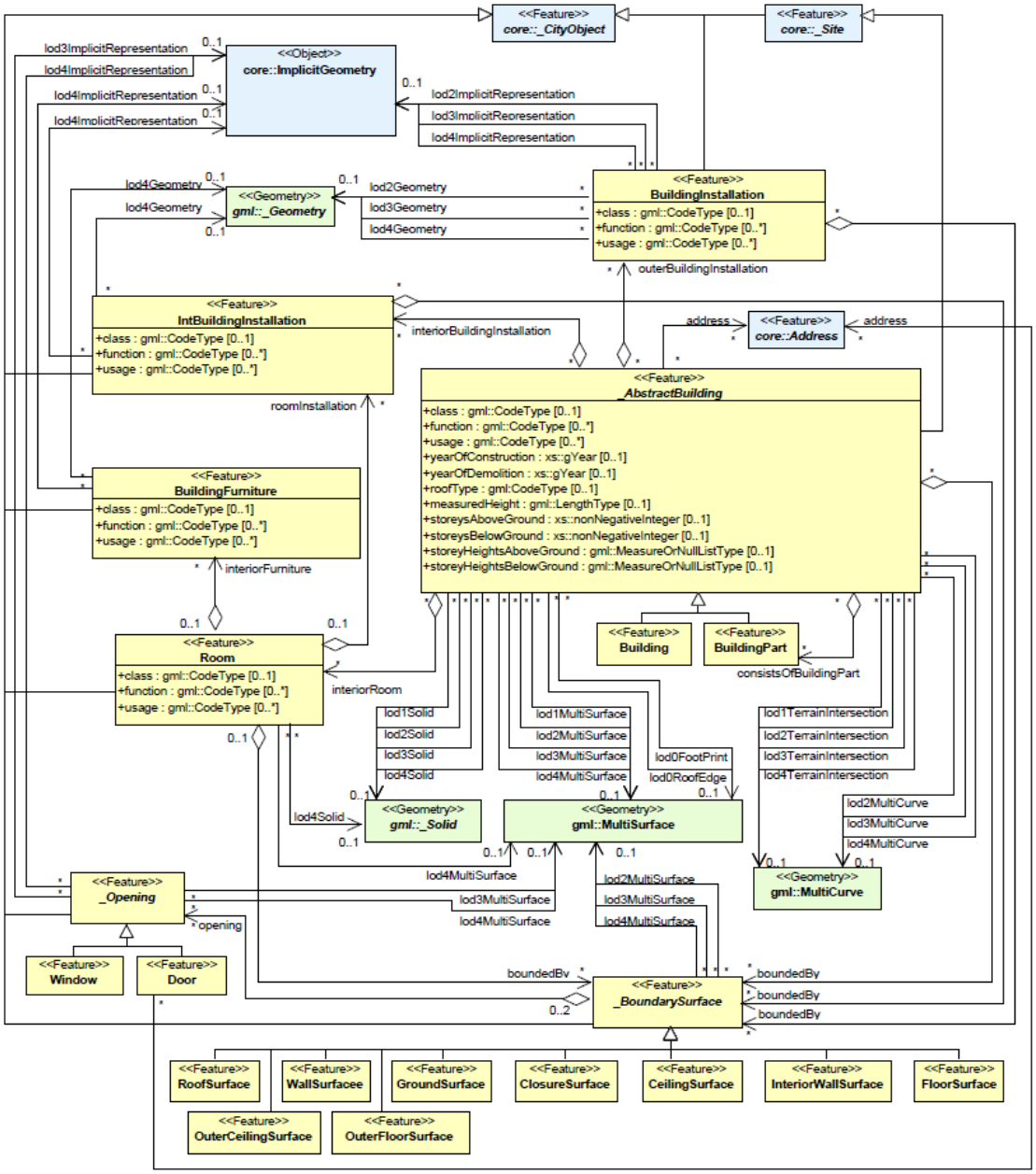

Figure 3.10.: CityGML building module. Source: [Gröger et al., 2012]

Figure 3.11.: Example of buildings consisting of building parts. Source: [Gröger et al., 2012]

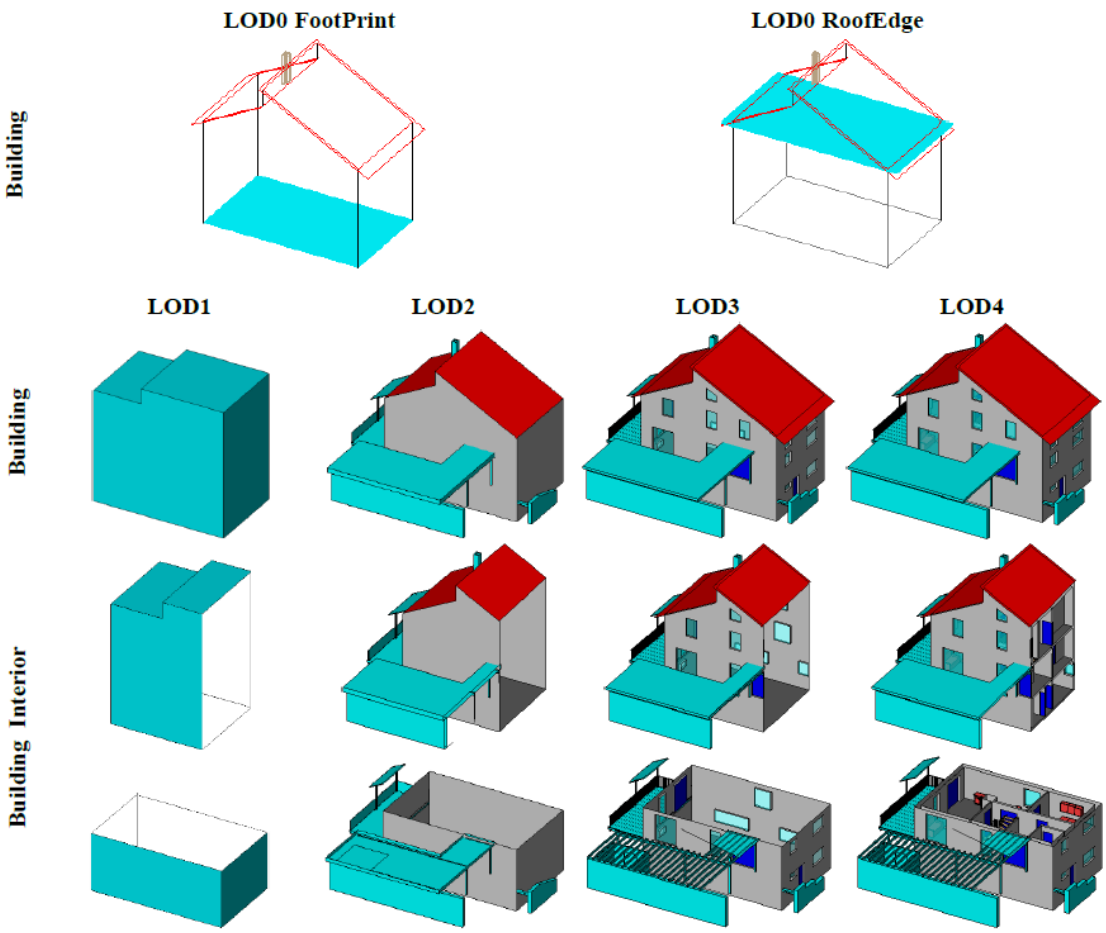

Figure 3.12.: Levels of detail of the building model of CityGML. Source: [Gröger et al., 2012]

4. Energy ADE

4.1. Introduction

This section contains a general introduction to the CityGML Energy Application Domain Extension[1] (Energy ADE) which utilizes the CityGML mechanism for extension (see chapter 3 section 3.3.5). At first a short overview will be given, followed by the main characteristics.

4.2. General

CityGML is being used and adopted more and more for applications in different scientific fields as a integrated and coherent information hub. When it comes to buildings, for example, CityGML can already store attributes like the year of construction, the building class and use. However, the possibility to natively store all relevant energy-related features and attributes in a systematic and standard way is not given [Agugiaro, 2016a].

The CityGML concept of Application Domain Extensions (ADE) provides a mechanism to simply add new features and/or properties extending the given modelling capabilities (see section 3.3.5 for details). In May 2014 an international consortium of urban energy simulation developers and users started to develop an Energy ADE to allow representation, storage and exchanging of energy-related features and attributes relevant to urban energy models. A first release was made available in February 2015 (see https://github.com/cstb/citygml-energy). Its goal is to cope with heterogeneous data of different qualities, levels of details and complexities regarding urban energy models (eg. going from monthly energy balance of ISO 13790, to sub-hourly dynamic simulations like CitySim[2] or EnergyPlus[3]). It takes into consideration the INSPIRE Directive, as well as the recent US Building Energy Data Exchange Specification (BEDES) [Nouvel et al., 2015][NOUVEL et al., 2015].

[1] https://github.com/cstb/citygml-energy
[2] http://citysim.epfl.ch/
[3] https://energyplus.net/

4.3. Main characteristics

The Energy ADE is structured to be modular, in order to offer other domains reusability or extensions of certain modules (eg. module Occupancy for socio-economic applications, module Construction and Materials for acoustics or statics, etc.)[url, 2016d].

Energy Ade is currently consisting of 5 modules:

- Building Physics module
- Occupancy module
- Construction and Materials module
- Energy System module
- Timeseries and Schedules module

In the following sections the Energy ADE UML diagrams, the principal classes and their characteristics will be are presented and described in more detail.

4.3.1. Building Physics module

The Building Physics module serves as the core module of the Energy ADE (see fig. 4.1). Its main purpose is to model thermal properties. It extends the existing CityGML classes _AbstractBuilding, _BoundarySurface and _Opening resulting in new thermal classes like ThermalZone, ThermalBoundary, ThermalComponent. The CityGML _AbstractBuilding class (see fig. 4.1) is extended with a number of attributes like atticType and basementType (conditioned/unconditioned), the buildingType (for example apartment block, multi family house, etc.), the constructionStyle, different types of floor area (NetFloorArea, GrossFloorArea, EnergyReferenceArea), refurbishment measures (RefurbishmentMeasureBuilding) and energy performance certifications (EnergyPerformanceCertification) [url, 2016d]. The _BoundarySurface and _Opening classes (see fig. 4.1) are correspondently extended with other attributes. The CityGML class _BoundarySurface for example is extended with refurbishment measures (refurbishmentMeasure), globalSolarIrradiance (which is the sum of the direct, diffuse and reflected irradiance incident on a outside boundary surface in W/m^2) and daylightIlluminance (which is the sum of the direct, diffuse and reflected solar illuminance incident on a outside boundary surface in lx) [url, 2016d]. The most representative class of the building physics module is the ThermalZone (see fig. 4.1), which allows to model the thermal hull of a building [url, 2016d]. A thermal zone is defined as "a "thermal homogeneous" space considered as isothermal, but may also refer to several building rooms and zones with different usage boundary conditions for simplified building energy modelling [url, 2016d]". A thermal zone is bounded by a number of ThermalBoundary objects. These can either be semantically defined and linked to CityGML _BoundarySurfaces or modeled explicitly via the surfaceGeometry property[cf. url, 2016d].

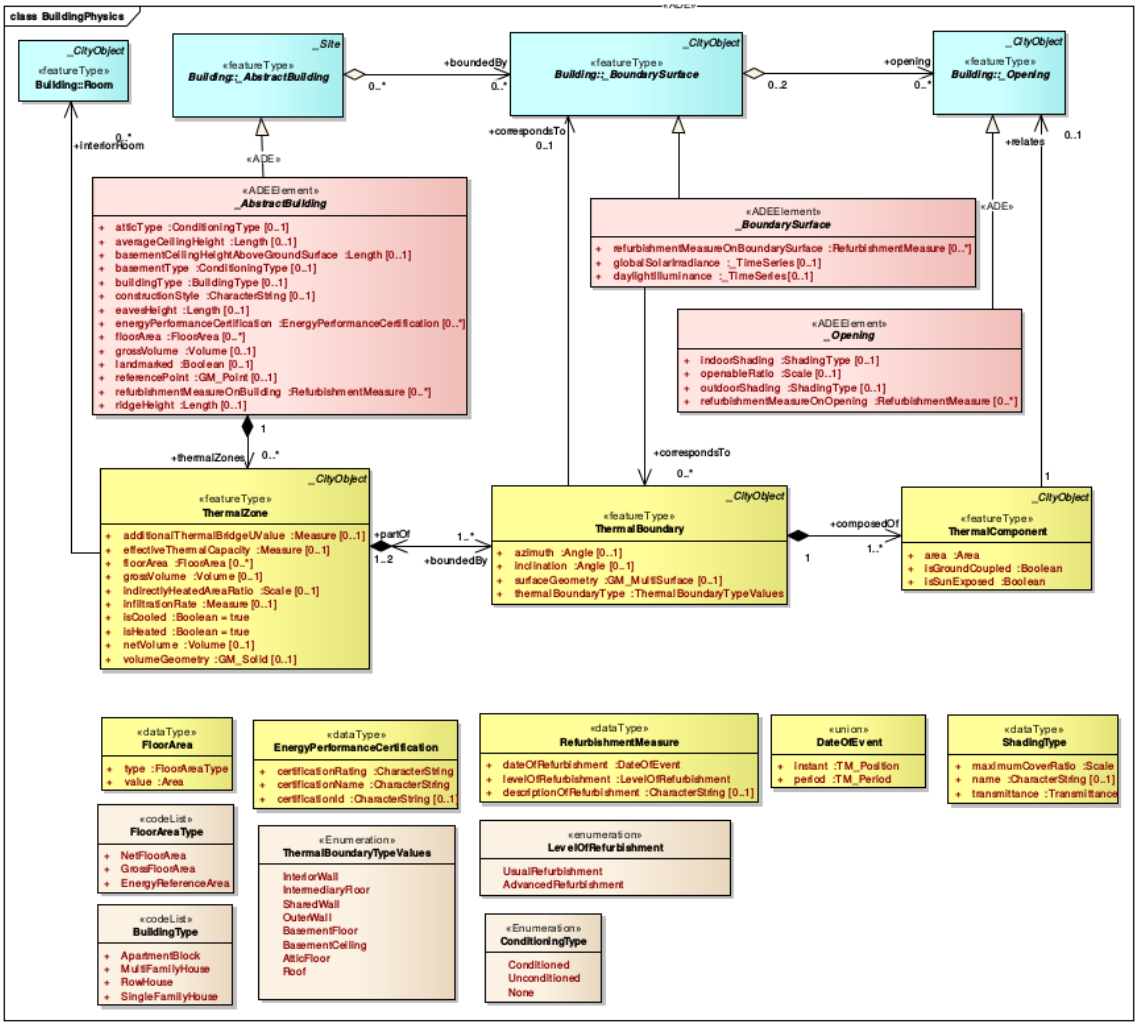

Figure 4.1.: UML diagram of the Building Physics module in the Energy ADE v. 0.6. Source: [url, 2016d]

4.3.2. Occupancy module

The Occupancy module (see fig. 4.2) focuses on the characterization of buildings in terms of usage, occupancy, and facilities. Its most representative class, the UsageZone, relates both to _AbstractBuilding and ThermalZone. A UsageZone is a zone of buildings with homogeneous usage conditions and indoor climate control settings. It is a semantic object, with an optional geometry (volumeGeometry), which may or may not be related to a geometric entity (Building, BuildingPart, Room etc.). One or more UsageZones are contained in a ThermalZone. Each UsageZone is characterized by attributes to describe its usage, schedules for Heating, Ventilation and Air Conditioning (HVAC), internal gains, and floor area values. UsageZones can contain BuildingUnits, which are a class introduced to define the ownership types of a single usage zone for building usage analyses or energy demand calculation. A BuildingUnit is a part of a single UsageZone which can be defined as a subdivision of a Building with its own lockable access from the outside or from a common area (i.e. not from another BuildingUnit). It is atomic, functionally independent, and may be separately sold, rented out, inherited. This definition corresponds with the one given by the INSPIRE Data Specification for Buildings. A BuildingUnit is related to one or more Occupant entities, such as a dwelling or a workplace. Owner information attributes are specified in this class. Facilities are any kind of devices which dissipate heat and should be accounted for in the energy demand calculation of a zone. Each UsageZone or BuildingUnit object can have one or more Facilitiesobjects. There are three subclasses of facilities for domestic hot water production, electrical appliances and lighting [url, 2016d].

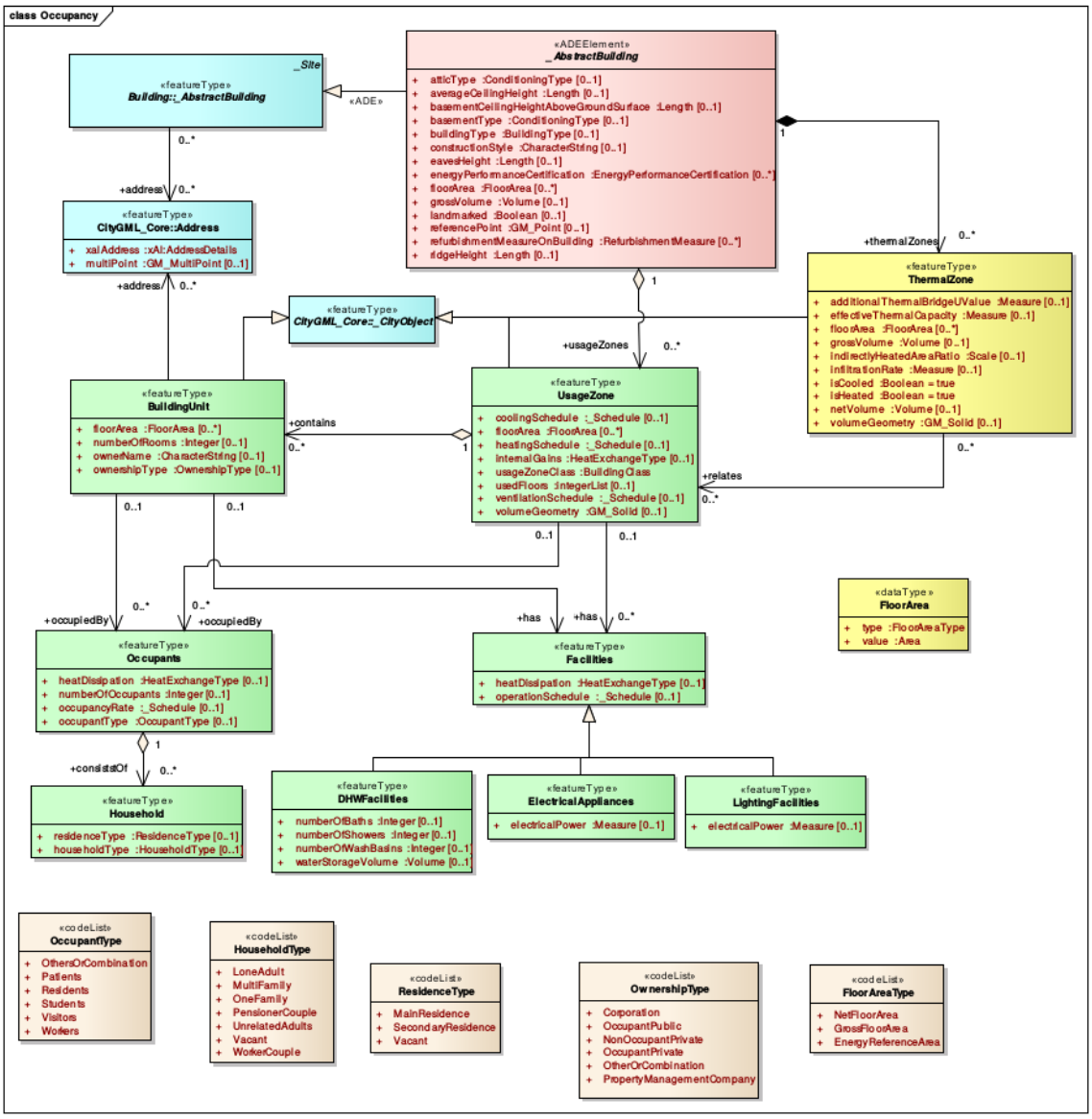

Figure 4.2.: UML diagram of the Occupancy module in the Energy ADE v. 0.6. Source: [url, 2016d]

4.3.3. Construction and Materials module

The Construction and Materials module contains the physical characterization construction elements of the building and can be applied to any CityGML _CityObject. It is applicable either to simply store U-values and g-values for walls and windows, but also to model more complex, multi-layered construction elements and their physical properties. In the Energy ADE, its main use is to characterise the geometrical surfaces of a building (RoofSurfaces, WallSurfaces, GroundSurfaces), but also the ThermalBoundaries of a ThermalZone [url, 2016d].

4.3.4. Energy Use and System module

The Energy System module (see fig. 4.3) defines the energy types (energy demand and sources) and energy systems (conversion, distribution and storage systems) to perform energy demand and supply analyses [url, 2016d].

The EnergyDemand class is the main object of the module and can be associated to any _CityObject (e.g. _AbstractBuilding, ThermalZone, UsageZone, BuildingUnit, etc.). On that account it represents the connection of the module to the rest of the Energy ADE and CityGML model. It describes the energy required to satisfy a specific end-use (space heating/cooling or domestic hot water) of a specified _CityObject. EnergyDemand is further described by the attributes energyAmount (time-depending energy demand values) and maximumLoad (yearly maximum load) [url, 2016d].

The energy use and system module defines classes both for energy storage (see class _StorageSystem in fig. 4.3) and distribution (see class EnergyDistributionSystem in fig. 4.3) within the building. Power and thermal distribution systems are differentiated (see classes ThermalDistributionSystem and PowerDistributionSystem in fig. 4.3). They all share a common distributionPerimeter attribute. Each EnergyDemand can have at maximum one EnergyDistributionSystem [url, 2016d]. A similar concept applies to energy storage (see classes _StorageSystem and PowerStorageSystem and ThermalStorageSystem, respectively in fig. 4.3). The class EnergyConversionSystem (see fig. 4.3) is used to model systems converting an energy source into the energy necessary to satisfy the EnergyDemand (or to feed the networks). It is the central element describing energy conversion. EnergyConversionSystem have common parameters regarding technical properties such as efficiencyIndicator, installedNominalPower, nominalEfficiency and other properties like the year of manufacture or refurbishment measure carried out on the EnergyConversionSystem object. An EnergyConversionSystem is further specialised in several other subclasses (SolarThermalSystem, PhotovoltaicSystem, HeatPump, DistrictNetworkSubstation, etc.), each with specific characteristics. In particular, the subclass DistrictNetworkSubstation contains attributes about the network ID and node ID and represents therefore the linkage to the outer world, i.e. outside the building, e.g. with respect to heating or cooling networks substations [url, 2016d].

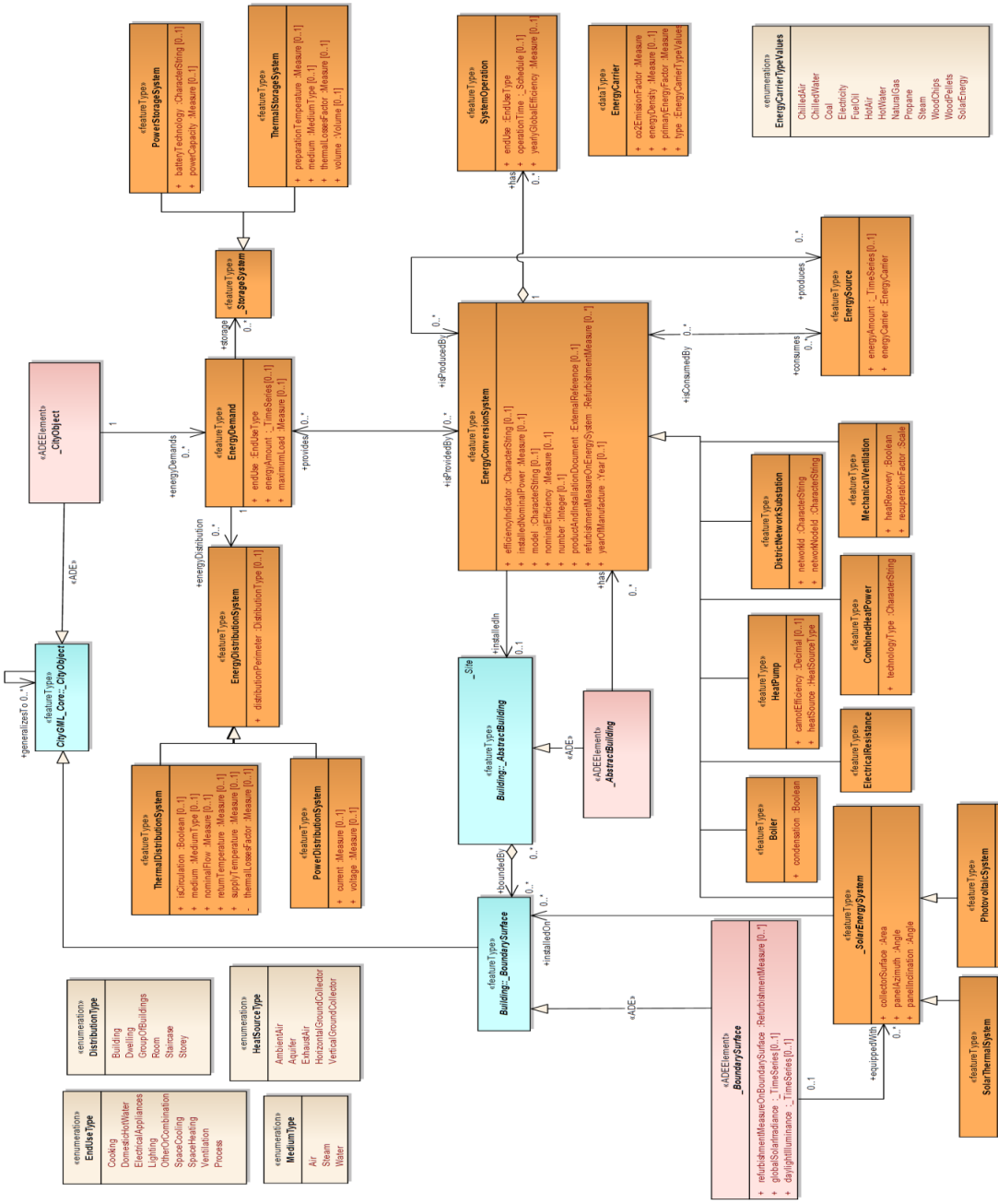

Figure 4.3.: UML diagram of the Energy module in the Energy ADE v. 0.6. Source: [url, 2016d]

4.3.5. Timeseries and Schedules module

All Energy ADE modules depend on a common module, the Temporal Data module, in which time-dependent variables (regular and irregular time series) and different types of schedules are defined as data types to be used by all Energy ADE elements. The inclusion of temporal variables, and the definition of their relevant properties such as the acquisition method (e.g. simulation or metering) allow to model and store both simulation results and metering data. Moreover, different time steps, corresponding to different building simulation methods and metering systems can be used.

5. EnergyPlus

5.1. Introduction

This chapter provides a general introduction and overview to the energy simulation program EnergyPlus[1]. Energy simulation can be described as using computer-based programs to model the energy performance of an entire building or the systems within a building which provides valuable information about the building and system energy use [JAWA, 2016]. At first a short overview about its origins and overall features is given followed by a description of EnergyPlus input files and available output formats.

5.2. General

EnergyPlus is a whole building energy simulation program, that engineers, architects and researchers use to model energy and water use in buildings. It originates from the BLAST (Building Loads Analysis and System Thermodynamics) and DOE-2 programs funded by the U.S. Department of Defense and the U.S. Department of Energy respectively [Crawley et al., 2001, JAWA, 2016].

Both were developed and released in the late 1970s as energy and load simulation tools, primarily addressing design engineers or architects to develop different energy related analyses. EnergyPlus is an energy analysis and thermal load simulation program, much like its parent programs. It calculates the heating and cooling loads necessary to maintain thermal control setpoints, the energy consumption of plant equipment as well as many other simulation details according to the needs of a designer. EnergyPlus shares many of the simulation characteristics with its parent programs BLAST and DOE-2 [Tzoulis, 2014, Dep, 2015b].

FORTRAN90 was chosen by the developers as the operating programming language for the initial release of EnergyPlus because of its advantages. Since the original version released in 2001, EnergyPlus code and structure continues to evolve and adopts Fortran Standard. With the release of EnergyPlus 8.2 in 2014 however, the development has been switched to C++ language in favour of a better built-in features for software development, testing, and maintenance. Figure 5.1 shows the overall structure, highlighting the main focus on the modular architecture [Dep, 2015b].

[1] https://energyplus.net/

New versions of EnergyPlus are published regularly. EnergyPlus does not have a user interface. It is intended to be the simulation engine which can also be complemented by a third-party interface [Tzoulis, 2014]. Google Sketch-Up[2] and integrated Open Studio[3] provide a graphical user interface for a more user friendly view and simplified input definitions. Google Sketch-Up is used for drawing buildings and envelope details (such as windows, doors or shadings), whereas all parameters can be defined via the Open Studio interface.

The following is a representative list of EnergyPlus capabilities, as given by the U.S. Department of Energy [Dep, 2015b]:

- Integrated, simultaneous solution where the building response and the primary and secondary systems are tightly coupled (iteration performed when necessary)
- Sub-hourly, user-definable time steps for the interaction between the thermal zones and the environment; variable time steps for interactions between the thermal zones and the HVAC systems (automatically varied to ensure solution stability)
- Heat balance based solution technique for building thermal loads that allows for simultaneous calculation of radiant and convective effects at both in the interior and exterior surface during each time step
- Transient heat conduction through building elements such as walls, roofs, floors, etc. using conduction transfer functions
- Improved ground heat transfer modeling through links to three-dimensional finite difference ground models and simplified analytical techniques
- Combined heat and mass transfer model that accounts for moisture adsorption/desorption either as a layer-by-layer integration into the conduction transfer functions or as an effective moisture penetration depth model (EMPD)
- Thermal comfort models based on activity, inside dry bulb, humidity, etc.
- Anisotropic sky model for improved calculation of diffuse solar on tilted surfaces
- Advanced fenestration calculations including controllable window blinds, electrochromic glazings, layer-by-layer heat balances that allow proper assignment of solar energy absorbed by window panes, and a performance library for numerous commercially available windows
- Daylighting controls including interior illuminance calculations, glare simulation and control, luminaire controls, and the effect of reduced artificial lighting on heating and cooling

[2]http://www.sketchup.com/
[3]https://www.openstudio.net/

- Atmospheric pollution calculations that predict CO2, SOx, NOx, CO, particulate matter, and hydrocarbon production for both on site and remote energy conversion

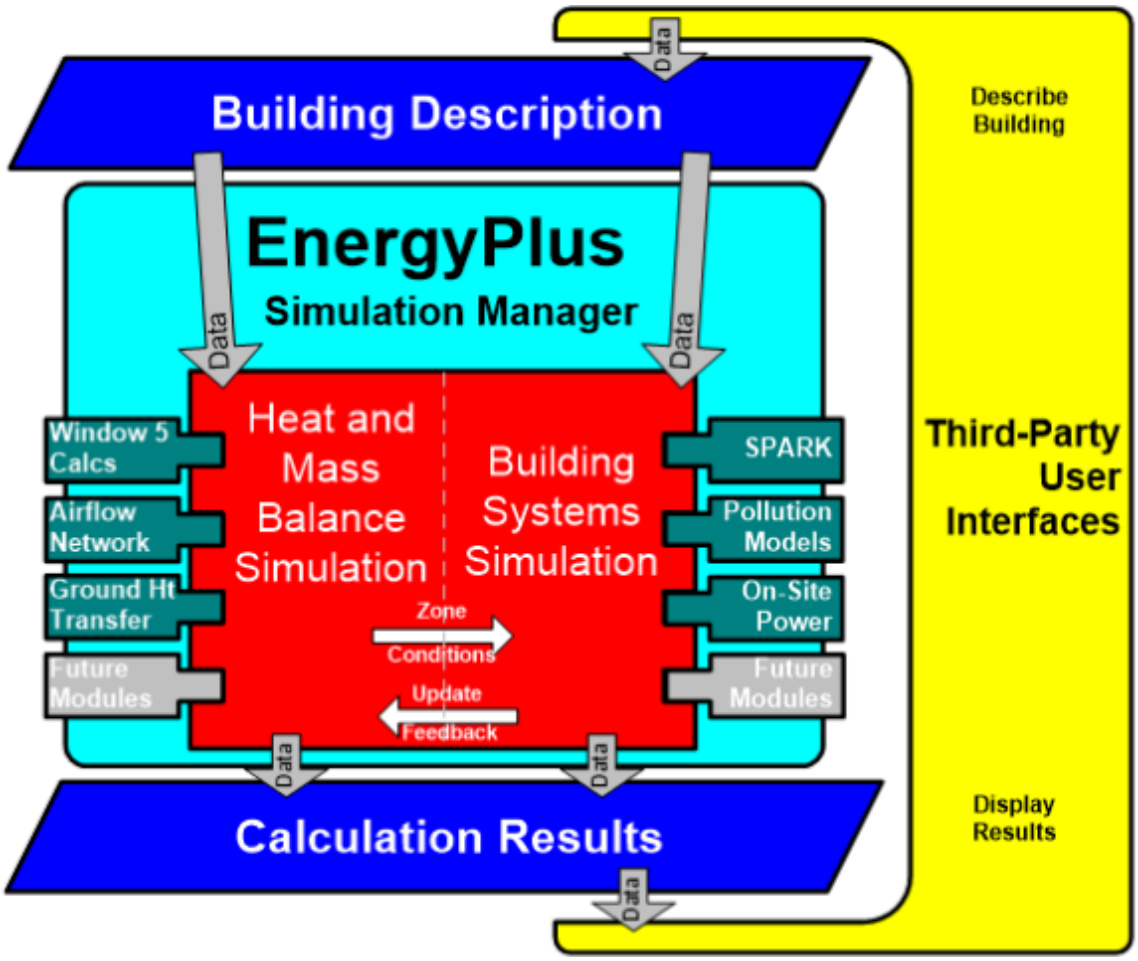

Figure 5.1.: Program structure of EnergyPlus. Source: [Crawley et al., 2001]

It needs to be emphasized, that different from the fixed time step used by most programs, EnergyPlus has sub-hourly, user-definable steps for the interaction between the thermal zones and the environment, and variable time steps for interactions between the thermal zones and the HVAC systems [Zhou, 2014]. The development team behind EnergyPlus tried to create a program that is relatively simple to work with from the perspective of both the users and the developer [Dep, 2015b].

5.3. Input and Output

Energy Plus requires 2 input files prior in order to successfully run a simulation, namely the Input Data Dictionary (IDD) and the Input Data File (IDF).

The Input Data Dictionary (IDD) defines the schema format and contains the definitions to all objects EnergyPlus will recognize and use during runtime (in fact

EnergyPlus will read all objects defined in the Input Data Dictionary (IDD) prior to running a simulation). It specifies all modules, their objects and the syntax requirements to be used for a simulation [Dep, 2015a,b, Banfi, 2013]. Input Data Dictionary (IDD) syntax can be structured into a hierarchy of 3 levels namely module, object and attribute. Module at the top level of the hierarchy structures the schema definition into certain range of topics (for example Air Distribution, Internal Gains, Schedules or even Simulation Parameters). Each module thematically groups together a series of objects which are furthermore described by certain attributes [Banfi, 2013]. At the time of writing the current version's (EnergyPlus 8.3) schema definition consists of 59 modules and over 1600 objects.

Listing 5.1 shows parts of the Input Data Dictionary (IDD) Building object definition.

The Input Data File (IDF) on the other hand contains the simulation data, for example the description of the building and the corresponding HVAC system. It is intended to conform to the schema definition in the Input Data Dictionary (IDD). Energy Plus will validate an Input Data File (IDF) based on the objects defined in the Input Data Dictionary (IDD) prior to running a simulation. Since each new release of EnergyPlus may be shipped with an extended or modified schema definition, legacy building models written in Input Data Files (IDFs) may be rendered incompatible.

Both the Input Data Dictionary (IDD) and the Input Data File (IDF) need follow a certain set of rules. General rules applying to both of them for example include: definitions must end with a semicolon, object fields must be comma-separated, each object must have a unique name, input records can be up to 500 characters in lengths, section and object keywords can be up to 100 characters in lenghts, special characters should not be included in the file, etc. [Dep, 2015a]. Both the Input Data Dictionary (IDD) and the Input Data File (IDF) are simple ASCII text files which can be inspected and modified using conventional text editors. See the "Energy Plus Interface Developer Guide[4]" and the "Energy Plus Getting Started[5]" for further detailed information.

Weather data is another major (but optional) input data in EnergyPlus. Its a simple text-based file. Weather data files include location information such as: data source, latitude, longitude, time zone, location, elevation, peak heating and cooling design conditions, typical and extreme periods, holidays, daylight savings period, and period covered by the data. Typically weather data files stretch over a full year (8760 or 8784 hours) altough EnergyPlus allows for subsets of years as well [Crawley et al., 2004].

EnergyPlus saves results for each time step during simulation. Reports can be requested either detailed containing each time step or more general with aggregated time intervals. EnergyPlus supports output to text files (comma/tab/space separated), HTML files for viewing in the browser or to SQLite[6] database files.

[4]http://bigladdersoftware.com/epx/docs/8-3/interface-developer/index.html
[5]http://bigladdersoftware.com/epx/docs/8-3/getting-started/index.html
[6]https://sqlite.org/

```
Building,
\memo Describes parameters that are used during the simulation
\memo of the building. There are necessary correlations between the entries for
\memo this object and some entries in the Site:WeatherStation and
\memo Site:HeightVariation objects, specifically the Terrain field.
\unique-object
\required-object
\min-fields 8
A1 , \field Name
\required-field
\retaincase
\default NONE
N1 , \field North Axis
\note degrees from true North
\units deg
\type real
\default 0.0
A2 , \field Terrain
\note Country=FlatOpenCountry | Suburbs=CountryTownsSuburbs | City=CityCenter |
    Ocean=body of water (5km) | Urban=Urban-Industrial-Forest
\type choice
\key Country
\key Suburbs
\key City
\key Ocean
\key Urban
\default Suburbs
N2 , \field Loads Convergence Tolerance Value
\note Loads Convergence Tolerance Value is a fraction of load
\type real
\minimum> 0.0
\maximum .5
\default .04
N3 , \field Temperature Convergence Tolerance Value
\units deltaC
\type real
\minimum> 0.0
\maximum .5
\default .4
```

Listing 5.1: IDD - Definition of the Building object

```
!- =========== ALL OBJECTS IN CLASS: BUILDING ===========

Building,
TemplateFull, !- Name
0.00, !- North Axis {deg}
City, !- Terrain
0.04, !- Loads Convergence Tolerance Value
0.4, !- Temperature Convergence Tolerance Value {deltaC}
FullInteriorAndExterior, !- Solar Distribution
90, !- Maximum Number of Warmup Days
6; !- Minimum Number of Warmup Days
```

Listing 5.2: IDF - Building object

6. Implementation of the data interface

6.1. Introduction

This chapter describes the implementation details of the bidirectional data interface between CityGML and EnergyPlus. At first a short overview of the development environment used is given (see section 6.2), followed by a description of the overall workflow of the data interface (see section 6.3). Section 6.4 covers aspects of the mapping process between CityGML and EnergyPlus classes. Section 6.5 gives a more detailed overview of the implementation details. Finally the restrictions and constraints encountered during the implementation of the data interface are discussed (see section 6.6).

6.2. Development Environment

This section gives a short overview and description of the development environment chosen.

Java

The data interface is developed in the Java[1] language developed by Sun Microsystems (now owned by Oracle Corporation) in version 7. Java is a general-purpose, concurrent, class-based and object-oriented language. It is a strongly typed, relatively high-level language which includes automatic storage management using a garbage collector implementation. The Java programming language is compiled to an intermediary bytecode which runs on the Java Virtual Machine (JVM) [Gosling, 2000]. Java bytecode can run on any Java Virtual Machine (JVM) regardless of the underlying computer architecture ("write once, run anywhere"). According to the TIOBE index[2] the Java Language was the most used programming language as of 2016.

[1] http://www.oracle.com/technetwork/java/javase/overview/index.html
[2] http://www.tiobe.com/tiobe-index/

Citygml4j

Citygml4j[3] is an open source Java class library and API for facilitating work with CityGML developed at the Institute for Geodesy and Geoinformation Science, Berlin University of Technology. The goal of citygml4j was to make it easy to read, process and write CityGML datasets and to develop CityGML-aware software applications. The library is released under the Apache License Version 2.0[4]. Citygml4j utilizes Java Architecture for XML Binding (JAXB)[5] to generate a set of Java classes that represent the CityGML schema definition. CityGML instance documents are deserialized into a tree of Java objects the developer can work with [cit, 2016].

Ade-xjc

Ade-xjc[6] is a command line tool and wrapper for the Java XML Schema binding compiler (xjc) shipped with Java Architecture for XML Binding (JAXB) making it possible to compile arbitrary CityGML Application Domain Extension (ADE) schemas to a set of corresponding JAXB classes to be used with citygml4j. The ade-xjc command line tool is released under the Apache License Version 2.0 [ade, 2016].

Enterprise Architect

Enterprise Architect[7] by Sparx Enterprise is a visual UML modeling and design tool. Trough its Scripting API it is possible to manipulate elements, diagrams and other model artifacts programmatically. [Banfi, 2013] used the Scripting API to automatically generate a UML model out of the EnergyPlus IDD (see chapter 5 section 5.3). Furthermore Enterprise Architect provides a mechanism to automatically generate source code in different languages directly from a UML model.

SketchUp

Sketchup[8] by Trimble is a easy to use 3D modeling software covering a wide range of drawing applications. It is both available as a freeware version (SketchUp Make) and a commercial one (SketchUp Pro). Since Version 4 SketchUp supports software extensions written in the Ruby Programming language.

[3]https://github.com/citygml4j/citygml4j
[4]http://www.apache.org/licenses/LICENSE-2.0
[5]http://www.oracle.com/technetwork/articles/javase/index-140168.html
[6]https://github.com/citygml4j/ade-xjc
[7]http://www.sparxsystems.com/products/ea/
[8]http://www.sketchup.com/

OpenStudio

OpenStudio[9] by the U.S. Department of Energy (DoE) is a cross-platform collection of software tools to support building modeling in EnergyPlus. Currently the OpenStudio distribution consists of the SketchUp plug-in, the OpenStudio Application, the Parametric Analysis Tool and the ResultViewer. Most notably the SketchUp plug-in provides an easy way to quickly import EnergyPlus IDFs (see chapter 5) into SketchUp to view or edit building geometries.

6.3. Workflow description

This section will give an overview regarding the workflow of the data interface implementation. The goal is to implement an interface able to transfer data bidirectionally between CityGML (and its Energy ADE) and EnergyPlus. More specifically the main tasks, that need to be carried out are the extraction of the CityGml's building geometries and all energy-relevant attributes and features and the mapping into the appropriate EnergyPlus counterpart in order to successfully complete energy demand simulations. Subsequently the simulation results need to be translated back into CityGML.

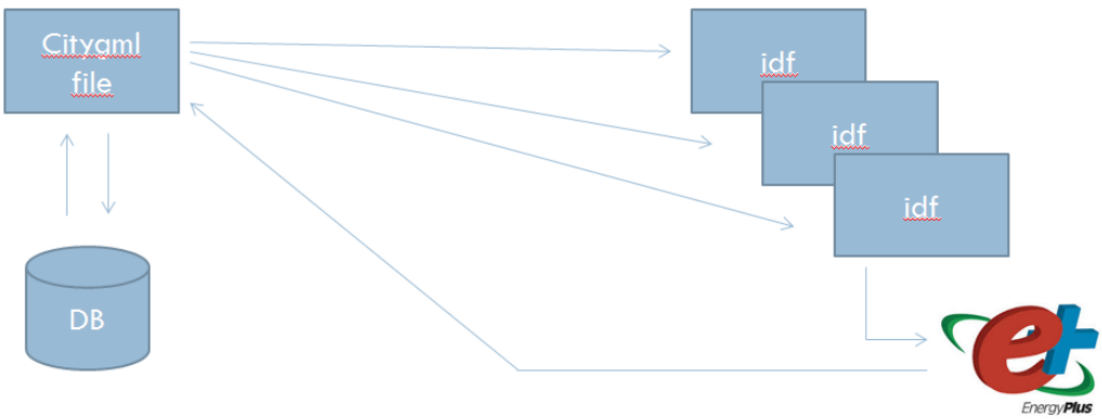

Figure 6.1.: General Workflow.

As figure 6.1 depicts, the general workflow consists of multiple steps. First of a CityGML instance document needs to be created, containing one or multiple buildings. Since EnergyPlus is a whole-building simulation software used to model energy performance of *single* buildings (as discussed in chapter 5) multiple Input Data Files (IDFs) need to be created. Each one of the newly created Input Data Files (IDFs) will then be imported into EnergyPlus and the actual energy simulation started. Lastly the results need to be written back to the CityGML file.

[9] https://www.openstudio.net/

6.4. Mapping between CityGML and EnergyPlus

In order to accomplish the task of comprehensively translating CityGML instance data to EnergyPlus Input Data Files (IDFs) a process called data mapping needs to be applied. Data mapping "is the process of creating data element mapping between two distinct data models" [Dong, 2007]. The first step is to compare both data models and identify commonalities, namely all relevant classes. Subsequently data transformation rules can be defined between those classes. EnergyPlus is a very complex simulation tool with regard to the data model (59 modules and over 1600 objects) allowing for a very detailed description of building models (see chapter 5). In order to keep within reasonable bounds and still produce meaningful simulation results a fraction of EnergyPlus objects were selected. Below an overview of the relevant EnergyPlus modules and their corresponding objects identified is given:

- Module *Simulation Parameters*:

  This module bundles the more general objects, which influences the simulation in various ways. *Version* indicates, which Energy Plus version the Input Data File (IDF) was created for. *SimulationControl* allows the user to specify what kind of calculations will be performed. *Building* describes parameters used during the simulation of the building like terrain or solar distribution. *SolarCalculation* controls some details of EnergyPlus's solar, shadowing and daylighting models. *SurfaceConvectionAlgorithm:Inside* and *SurfaceConvectionAlgorithm:Outside* control the models used for surface convection on the inside and outside. *HeatBalanceAlgorithm* defines the heat and moisture transfer algorithms. *Timestep* specifies the driving timestep for heat transfer and load calculations. The value entered indicates the number of timesteps within an hour [Dep, 2015c].

- Module *Location - Climate - Weather File Access*

  This module bundles objects, which describe the ambient conditions for the simulation. *Site:Location* describes the buildings location. A eventual existing weather data file location will override this objects parameters. *RunPeriod* is necessary to create a weather file simulation. *Site:GroundTemperature:BuildingSurface* is used for the ground heat transfer model and influences all building surfaces which touch the ground [Dep, 2015c].

- Module *Schedules*

  This module lets the user define schedules which influence the scheduling of many items such as occupancy, lighting, thermostatic controls and so on. *Schedule:Compact* and *ScheduleTypeLimits* allow for flexible definitions of schedules [Dep, 2015c].

- Module *Surface Construction Elements*

  This module describes the physical properties of the building like walls, roofs, floors, windows and doors. *Construction* and the corresponding *Mate-*

rial objects describe how walls, roofs and doors are built from the included materials. Materials for glass windows and doors can be described using the objects *WindowMaterial:SimpleGlazingSystem* [Dep, 2015c].

- Module *Thermal Zone Description/Geometry*

  This module bundles objects which describe the thermal zone characteristics and the details of each surface to be modeled. Of course thermal zones and surfaces are essential object without the building can't be simulated. *Zone* defines a thermal zone of the building. The object is identified via a property *Name* which subsequent *Surface* objects can reference to. Energy Plus allows for several different surface types. *BuildingSurface:Detailed* is used to describe walls, roofs, and floors. The object is described by *Name*, *Surface Type* (Ceiling, Floor Wall or Roof), their *Construction*, their corresponding *Zone*, their *Outside Boundary Condition* (adjacent to another surface, zone, outdoors or ground), *Sun Exposure* and *Wind Exposure* and their geometric representation. *FenestrationSurface:Detailed* is used for subsurfaces like Windows and Doors. This object inherits many properties of their corresponding base surface (e.g. *BuildingSurface:Detailed*). A *FenestrationSurface:Detailed* references a *Construction* and is described geometrically as well. *GlobalGeometryRules* offers a way to specify the geometric rules used to describe the input of surface vertices [Dep, 2015c].

- Module *Internal Gains*

  This module bundles objects that describe influences on energy consumption in the building other than envelope and ambient conditions (like people, lights, electric equipment). *People*, *Lights* and *ElectricEquipment*, *HotWaterEquipment* are used to model the effects of occupants, electric lighting systems, equipment consuming electricity (computers, televisions and so on) and hot water equipment consuming district heating in the zone [Dep, 2015c].

- Module *Airflow*

  The airflow between zones and due to natural ventilation (for example open windows) play an important role regarding energy consumption. *ZoneInfiltration:DesignFlowRate* allows for describing the unintentional flow of air form the outdoor environment directly into a thermal zone (like opening or closing of doors and windows or cracks around windows) [Dep, 2015c].

- Module *Design Objects*

  This module represents certain objects that are necessary for Energy Plus in order to successfully calculate zone design heating and cooling loads and air flow rates. The object *DesignSpecification:ZoneAirDistribution* describes the air distribution effectiveness of a zone. *DesignSpecification:OutdoorAir* is used to describe general outdoor air requirements which are referenced by other objects [Dep, 2015c].

- Module *Zone Controls - Thermostats and Humidistats*

This module bundles objects used to control zone conditions to a specific setpoint. *ZoneControl:Thermostat* and *ZoneControl:Humidistat* are used to control a zone to a specified temperature or to a specified relative humidity, respectively. *ThermostatSetpoint:DualSetpoint* is used for a heating and cooling thermostat with dual setpoints, where the setpoints can be scheduled troughout the simulation for both heating and cooling [Dep, 2015c].

- Module *Zone Equipment*

  This module provides objects to model HVAC (heating, ventilation and air conditioning) systems for a certain zone. *ZoneHVAC:EquipmentList* lists all HVAC equipment serving the zone. *ZoneHVAC:EquipmentConnections* defines the remaining details about each thermal zone from an HVAC perspective [Dep, 2015c].

- Module *Zone Forced Air Units*

  The simplest piece of zone equipment is the *ZoneHVAC:IdealLoadsAirSystem* component. It is used to study the performance of a building without modeling a full HVAC system [Dep, 2015c].

- Module *Reports*

  This module provides objects to configure what will appear in the output files. *Output:Variable* is used to request certain result reporting. Energy Plus offers a wide range of output variables available for reporting results, the most important being *"Surface Outside Face Solar Radiation Heat Gain Rate per Area"* (which describes the heat transferred by the absorption of solar radiation at the outside face being the result of incident solar radiation being absorbed at the surface face), *"Zone Ideal Loads Zone Total Cooling Rate"* (which describes the total (sensible and latent) cooling rate removed from the zone) and *"Zone Ideal Loads Zone Total Heating Rate"* (which describes the total (sensible and latent) heating rate added to the zone). *Output:SQLite* requests that the results need to be written to SQLite files for further processing [Dep, 2015c].

6.4.1. Example - Mapping of Energy Plus Material and CityGML module Construction

Figure 6.2 and table 6.1 depict a simpler mapping example between the CityGML Energy ADE module *Construction and Material* and the EnergyPlus equivalent object *Material*. The *Material* object field *Name* is a unique reference name, that is assigned to a particular material. It is expressed by the value of the GML element *gml:id*. This name can then be referred to by other input data. *Material* field *Roughness* defines the relative roughness of a particular material layer. This parameter influences the convection coefficients. As there is no equivalent CityGML counterpart, the default value of *MediumRough* is used. CityGML Energy ADE class *SolidMaterial* holds the attributes *Conductivity*, *Density* and *Thickness* that can be mapped to their EnergyPlus *Material* counterpart attributes of the same name. *Conductivity* describes the thermal conductivity of the material layer in W/mK. *Density* describes the density of the material layer in units of kg/m^3. *Specific Heat* represents the specific heat of the material layer in units of J/kgK, whereas only values of specific heat of 100 or larger are allowed. *Thermal Absorptance*, *Solar Absorptance* and *Visible Absorptance* fields are not mandatory and therefore left blank [Dep, 2015c].

Table 6.1.: Mapping between EnergyPlus Object Material and Energy ADE Construction and Material.

EnergyPlus Material	Energy ADE Construction and Material
Name	_CityObject:gml:id
Roughnes (default: MediumRough)	N/A
Thickness	LayerComponent:thickness
Conductivity	SolidMaterial:conductivity
Density	SolidMaterial:density
Specific Heat	SolidMaterial:specificHeat
Thermal Absorptance (not mandatory)	-
Solar Absorptance (not mandatory)	-
Visible Absorptance (not mandatory)	-

See Appendix A for a complete listing of all mappings between CityGML and EnergyPlus.

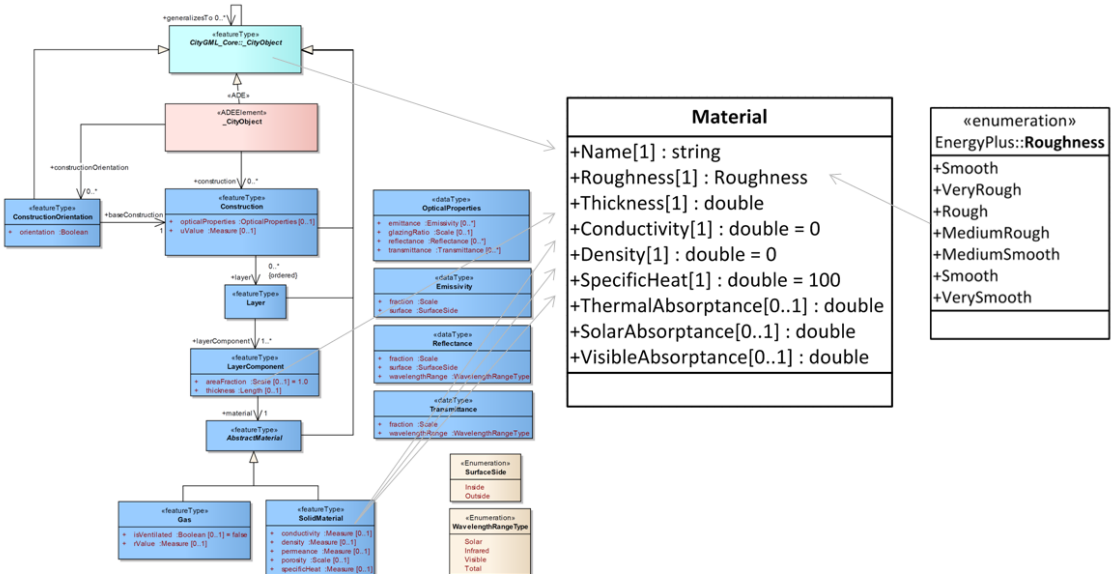

Figure 6.2.: Mapping example: CityGML Construction module to Energy Plus Materials object.

6.5. Implementation Details

Figure 6.3 gives a schematic design overview of the data interface. It consists of 3 different modules: Core, Invoker and ResultHandler.

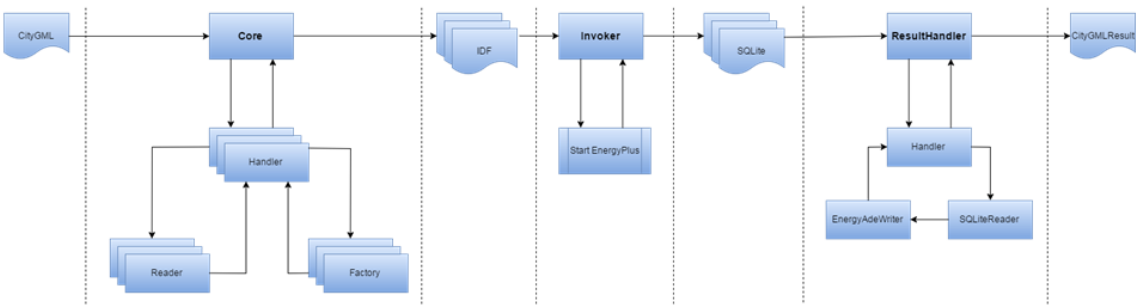

Figure 6.3.: Schematic high level design overview.

The Core module processes a CityGML instance document producing Input Data Files (IDFs) for each building. It deserializes the CityGML instance document into a tree of Java objects, traverses the tree and passes CityGML Building objects to a series of different handlers. These handlers encapsulate a series of readers in order to extract relevant data out of the building objects passed to them. The handlers further communicate with different factories, passing the extracted data. The factories are responsible for creating the associated EnergyPlus objects and mapping the data accordingly. Lastly the handlers create a string representation of all EnergyPlus objects to be written into a new Input Data File (IDF).

The Invoker module starts a Energy Plus simulation run for each of the provided IDFs. A successful simulation run yields results in the form of SQLite databases[10]. SQLite is a file based database which can easily be queried programmatically using Java Database Connectivity (JDBC)[11].

The ResultHandler module processes SQLite result databases appending relevant information to the input CityGML instance document. It uses SQL queries to extract the relevant data into a list of Java Objects which subsequently gets passed to an EnergyAdeWriter. The EnergyAdeWriter handles the process of serializing the Java objects to an enriched output CityMGL instance document.

Section 6.5.1, 6.5.2 and 6.5.3 provide a more detailed overview about the implementation of each module.

6.5.1. Core Module

As already mentioned in the introduction (see chapter 6.5) the Core module processes a CityGML instance document and produces one Input Data File (IDF) for each building. In order to do this a series of preliminary work is necessary. Sections 6.5.1.1 and 6.5.1.2 provide an overview of how work with ADE-enriched CityGML documents and EnergyPlus, respectively. The main topic of section 6.5.1.3 is the implementation of the core module, where everything is put together.

6.5.1.1. Handling CityGML and Energy ADE

Since CityGML is an XML-based standard (see chapter 3) the JAVA language already provides several methods for processing and writing XML which may introduce a lot of effort to work with. The citygml4j[12] library tries to abstract most of the complexity of working with an CityGML document. It is an open source Java class library and API for facilitating work with CityGML providing convenience methods to read, process and write CityGML datasets (see chapter 6.2).

Figure 6.4 shows an example of how to open and traverse a CityGML instance document using citygml4j. Citygml4j offers a mechanism to handle Application Domain Extensions (ADEs) as well. In Connection with *ade-xjc*[13] (see chapter 6.2), which is a command line tool and wrapper for the XML schema binding compiler *xjc*[14], it is possible to compile arbitrary CityGML Application Domain Extensions (ADEs) into corresponding JAXB classes to be used with citygml4j [ade, 2016]. To control the compilation process ade-xjc offers multiple program arguments (see *https://github.com/citygml4j/ade-xjc* for a complete listing).

[10]https://sqlite.org/
[11]http://www.oracle.com/technetwork/java/javase/jdbc/index.html
[12]https://github.com/citygml4j/citygml4j
[13]https://github.com/citygml4j/ade-xjc
[14]https://docs.oracle.com/javase/8/docs/technotes/tools/unix/xjc.html

```java
System.out.println(df.format(new Date()) + "setting up citygml4j context and JAXB builder");
CityGMLContext ctx = new CityGMLContext();
CityGMLBuilder builder = ctx.createCityGMLBuilder();
CityGMLInputFactory in = builder.createCityGMLInputFactory();
CityGMLReader reader = in.createCityGMLReader(new File("../../datasets/LOD2_Buildings_v100.gml"));

while (reader.hasNext()) {
    CityGML citygml = reader.nextFeature();

    if (citygml.getCityGMLClass() == CityGMLClass.CITY_MODEL) {
        CityModel cityModel = (CityModel)citygml;

        int count = 0;
        for (CityObjectMember cityObjectMember : cityModel.getCityObjectMember()) {
            AbstractCityObject cityObject = cityObjectMember.getCityObject();
            if (cityObject.getCityGMLClass() == CityGMLClass.BUILDING)
                System.out.println("Found building.")
        }
    }
}
```

Figure 6.4.: Example of reading a CityGML document using citygml4j. Source: [cit, 2016]

```
ph@ph-pc: ~/tmp/energyade
ph@ph-pc:~/tmp/energyade$ java -jar ade-xjc.jar xsd/energy_ade.xsd -output ./output -binding binding.xjb -package at.ac
.ait.citygml.ade.energy
[19:02:22 info] Starting ade-xjc compiler
[19:02:22 info] Setting up build environment
[19:02:22 info] Running in strict mode. Checking sanity of subfolder 'schemas'
[19:02:22 info] Using ADE schema /home/ph/tmp/energyade/xsd/energy_ade.xsd
[19:02:22 info] Using JAXB binding /home/ph/tmp/energyade/binding.xjb
[19:02:22 info] Using Java package at.ac.ait.citygml.ade.energy for JAXB classes
[19:02:22 info] Generating JAXB classes. This may take some time...
[19:02:26 info] JAXB classes successfully written to /home/ph/tmp/energyade/output
ph@ph-pc:~/tmp/energyade$
```

Figure 6.5.: Generating Energy ADE JAXB classes.

Figure 6.5 shows how to generate JAXB classes for the Energy ADE. The first argument defines the location of the Energy ADE schema definition file. The *output* parameter defines where put the generated classes. The third parameter *binding* defines the location of the optional global JAXB binding file, which is used to prevent naming collisions in the generation process. The fourth parameter *package* defines the name of the Java package to be used for the JAXB classes, in order to override the default value of *ade*.

```java
System.out.println(df.format(new Date()) + "creating JAXBContext from ADE JAXB classes");
String contextPath = JAXBContextPath.getContextPath("at.ac.ait.citygml.ade.energy");
JAXBContext jaxbCtx = JAXBContext.newInstance(contextPath);

final Binder<Node> binder = jaxbCtx.createBinder();

for(ADEComponent adeComponent : building.getGenericApplicationPropertyOfAbstractBuilding())
{
    Element adeElement = adeComponent.getContent();
    JAXBElement<?> element = (JAXBElement<?>)binder.unmarshal(adeElement);

    if (element.getValue() instanceof ThermalZonePropertyType)
    {
        ThermalZoneType thermalZone = ((ThermalZonePropertyType)element.getValue()).getThermalZone();

        [...]
    }
}
```

Figure 6.6.: Example of reading a ADE-enriched CityGML document using citygml4j. Source: [cit, 2016]

Figure 6.6 shows an example of how to handle ADE enriched CityGML files using JAXB classes generated by ade-xjc. Focus should be directed at figure 6.6 line

number 12 where the *JAXBElement* value is checked for being an instance of the JAXB generated class *ThermalZonePropertyType*.

6.5.1.2. Handling EnergyPlus

EnergyPlus is a very complex simulation tool consisting of over 1600 objects to describe buildings (see chapter 5), making it a very complex and error-prone task to translate this objects into corresponding Java classes to work with. As already discussed in chapter 1, [Banfi, 2013] showed a way to automatically create a UML diagram out of the EnergyPlus Input Data Dictionary (IDD) using Enterprise Architect's scripting API (see chapter 6.2). Enterprise Architect provides a mechanism to automatically generate source code to different languages directly from a UML model.

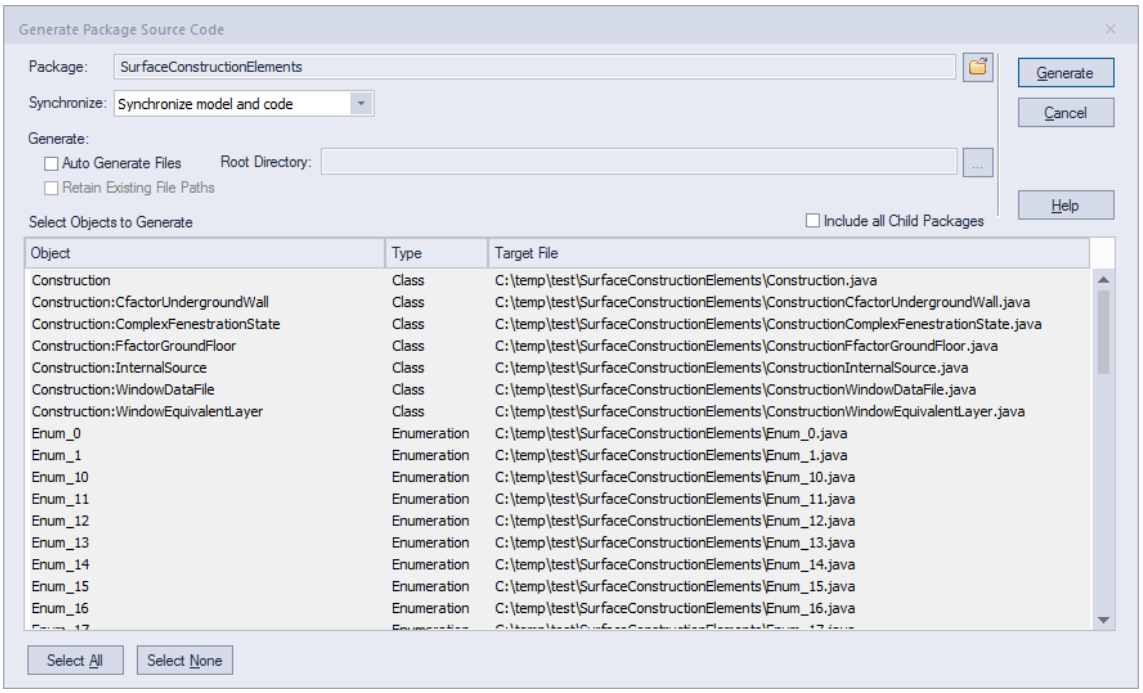

Figure 6.7.: Generating Java classes from EnergyPlus module SurfaceConstructionElements.

Figure 6.7 shows the Enterprise Architect dialog for generating Java classes from the EnergyPlus module *SurfaceConstructionElements*. The generated classes contain all the properties as defined in the EnergyPlus Input Data Dictionary (IDD). In the next step each of the generated Java classes got encapsuled in a so-called wrapper class. In object oriented programming, a wrapper class is a class that encapsulates types [Chapman, 1999] in order to add functionality like to generate the IDF representation of the object or to return the object name of the encapsulated type (see figure 6.8 for the interface every wrapper class needs to implement). The idea behind this mechanism was to be able to simply replace the generated EnergyPlus Java classes (in case of a new EnergyPlus version) and not having to rewrite the added functionality.

```
1  package at.ac.ait.energyplus;
2
3  public interface IEnergyPlus
4  {
5      public String toEnergyPlusString();
6      public String getEnergyPlusObjectName();
7  }
8
```

Figure 6.8.: Java interface IEnergyPlus.

6.5.1.3. Implementation

The main class of the core module is *CityGmlToEnergyPlusConverter*. It consists of one method named *convert* which expects a path to a valid CityGML document and a path to an output directory and produces a *CityGmlToEnergyPlusConverterResult* containing a list of the buildings processed (see figure 6.9).

```
1  public class CityGmlToEnergyPlusConverter {
2      public CityGmlToEnergyPlusConverterResult convert(File cityGml, File outputDirectory) {
3          [...]
4      }
5  }
```

Figure 6.9.: Core module class CityGmlToEnergyPlusConverter outline.

The file is first checked for its existence and then is deserialized into a tree of Java objects using the citygml4j library (as discussed in section 6.5.1.1). Each node of the tree is checked for being an instance of CityGML *Building* object, which, in case, gets further processed by being passed to a series of handlers, namely *BuildingPhysicsHandler*, *HVACTemplateHandler*, *MaterialHandler* and *OccupancyHandler*. Each one of the handler classes is, in essence, responsible for parsing a CityGML *Building* object into its EnergyPlus IDF representation. See figure 6.10 for the method signatures shared by each handler class.

```
1  public void process()
2  public String getEnergyPlusIDFString()
```

Figure 6.10.: Core module handler classes shared methods.

The method *process* is responsible for extracting the relevant information out of the CityGML *Building* object and subsequently calling appropriate factory methods in order to create the equivalent EnergyPlus object. The method *getEnergyPlusIDF-String* on the other hand, is responsible for creating a String representation of the previously created EnergyPlus objects, ready to be written to file.

Figure 6.11 exemplarily shows *BuildingPhysicsHandler* class implementation of the *process* method. In line 3 and 4 *ThermalZones* and *ThermalBoundaries* objects are read from the *Building* object. Afterwards each of the *ThermalZone* is passed to the *CityGmlEnergyPlusObjectFactory* to create its equivalent EnergyPlus counterpart, namely *ZoneWrapper* (line 8) which is subsequently added to a list (line 9) for further processing. Figure 6.12 shows the *createZone* method implementation of the

```
1   public void process()
2   {
3       HashMap<String, ThermalZoneType> tZones = reader.getThermalZones();
4       HashMap<String, ThermalBoundaryType> thermalBoundaries = reader.getThermalBoundaries();
5
6       for (ThermalZoneType tZone : tZones.values())
7       {
8           ZoneWrapper zWrapper = CityGmlEnergyPlusObjectFactory.createZone(tZone, building);
9           thermalZoneWrappers.add(zWrapper);
10      }
11
12      processThermalBoundarySurfaceGeometry(thermalBoundaries);
13  }
```

Figure 6.11.: Core module BuildingPhysicsHandler handler process method implementation.

CityGmlEnergyPlusObjectFactory class. In line 3 a new instance of the EnergyPlus *ZoneWrapper* object is created. The mapping part in this example simply consists of setting the id accordingly (line 5).

```
1   public static ZoneWrapper createZone(ThermalZoneType thermalZone, AbstractBuilding building) {
2       Zone zone = new Zone();
3       ZoneWrapper zoneWrapper = new ZoneWrapper(zone);
4
5       zone.setName(thermalZone.getId());
6
7       return zoneWrapper;
8   }
```

Figure 6.12.: Core module CityGmlEnergyPlusObjectFactory class creatZone method implementation.

As already mentioned, the method *getEnergyPlusIDFString* is responsible for accumulating the IDF String representations by calling the method *toEnergyPlusString* on each of the previously created EnergyPlus wrapper objects. Figure 6.13 shows the *getEnergyPlusIDFString* method implementation of the *BuildingPhysicsHandler* class.

```
1   public String getEnergyPlusIDFString()  {
2       StringBuilder sb = new StringBuilder();
3
4       for(ZoneWrapper zWrapper : thermalZoneWrappers)
5       {
6           sb.append(zWrapper.toEnergyPlusString());
7           sb.append(System.getProperty("line.separator"));
8       }
9
10      return sb.toString();
11  }
```

Figure 6.13.: Core module BuildingPhysicsHandler class getEnergyPlusIDFString method implementation.

Finally, after every handler class processed the CityGML *Building* object and generated the according EnergyPlus String representation, the Input Data File (IDF) is ready to be written to the output directory. The gml:id of the CityGML *Building* serves as the filename of the Input Data File (IDF).

6.5.2. Invoker Module

The Invoker module starts a Energy Plus simulation run for each of the previously generated IDF files. A successful simulation run yields results in the form of SQLite databases. The main class of the invoker module is *EnergyPlusInvoker*. The class is instantiated via its constructor with the String parameters *energyPlusDir* (which describes the path of the EnergyPlus installation directory), *inputDir* (which describes the path of the directory containing the IDF files), *idfFileNameWithoutExtension* (which is the name of the IDF file to be simulated by EnergyPlus), *outputDir* (which describes the path the EnergyPlus result files should be put) and *absolutWeather-DataFilePathWithExtension* (which describes the file location of the weather data file to be used during a simulation run). It furthermore consists of just one method named *invoke* (see figure 6.14) which creates a list of command line arguments in order to start a new process executing a EnergyPlus simulation run. Java

```
1  public class EnergyPlusInvoker {
2      public void invoke() throws EnergyPlusInvokerException
3  }
```

Figure 6.14.: Invoker module class EnergyPlusInvoker outline.

Standard EnergyPlus installation comes with a file called *Epl-run.bat*. This batch file can execute EnergyPlus using command line parameters. Listing 6.1 shows the syntax used.

```
1 Epl-run.bat <epin> <epout> <epinext> <epwthr> <eptype> <pausing> <maxcol> <
    convESO> <procCSV> <cntActv> <multithrd>
```

Listing 6.1: Syntax used to execute EnergyPlus via command line.

Parameter *epin* contains the file with full path and no extensions for input files, *epout* contains the file with full path and no extensions for output files, *epinext* contains the extension of the file (usually *idf*), *epwthr* contains the file with full path and extension for the weather file, *eptype* contains either "EP" or "NONE" to indicate if a weather file is used, *pausing* contains Y if pause should occur between major portions of batch file, *maxcol* contains an integer to limit columns or "nolimit" if unlimited, *convESO* contains Y if convertESOMTR program should be called, *procCSV* contains Y if csvProc program should be called, *cntActv* contains the count of other simulations active or about to be active and *multithrd* contains N if multithreading should be disabled.

The syntax used by the invoker module to execute EnergyPlus can be seen in listing 6.2. Parameters *epin*, *epout*, *epinext* and *epwthr* change accordingly.

```
1 Epl-run.bat <epin> <epout> <epinext> <epwthr> EP N NOLIMIT N N 0 Y
```

Listing 6.2: Syntax used by the invoker module to execute EnergyPlus.

The Java language provides a class called *ProcessBuilder*[15] which is used to create operating system processes. Each *ProcessBuilder* instance manages a collection

[15]https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html

of different attributes. Calling the *start* method creates a new process instance with those attributes [jav, 2016]. After a successful EnergyPlus simulation run the resulting SQLite database is copied to the output directory.

6.5.3. ResultHandler Module

The result handler module uses SQL queries to extract relevant data out of the SQLite result databases. After processing the relevant data is written back to the CityGML instance document. The main class of the invoker module is *ResultHandler*. The class is instantiated via its constructor with an instance of *ResultHandlerInputObj*, which encapsulates the attributes *cityGmlFile* indicating the location of the CityGML document, *workDirectory* indicating the directory path containing the previously generated SQLite files, *outPutFile* indicating an optional output file name and a list of the previously processed CityGML *Building* object ids. The *ResultHandler* class provides one public method named *process* (see figure 6.15).

```
1  public class ResultHandler {
2      public void process() throws CityGmlResultHandlerException
3  }
```

Figure 6.15.: Result handler method outline.

Figure 6.16 shows the implementation of the *process* method. After getting the connection to the current result SQLite file (line 1-3), the private methods *handleSolarRadiationHeatGain*, *handleCoolingRate* and *handleHeatingRate* are called. These methods query the databases for the output of the variables *"Surface Outside Face Solar Radiation Heat Gain Rate per Area"*, *"Zone Ideal Loads Zone Total Cooling Rate"* and *"Zone Ideal Loads Zone Total Heating Rate"* as requested in the Input Data File (IDF) (see section 6.4 for further information). Subsequently the private method *writeOutputObjs* is called, which is responsible for writing the collected data back to the CityGML document.

```
1  AbstractBuilding building = buildings.get(id);
2  File sqlLiteFile = new File(inputObj.getWorkDirectory(), id + ".sql");
3  Connection con = DbConnection.getSqlLiteConnection(sqlLiteFile);
4
5  handleSolarRadiationHeatGain(con, building);
6  handleCoolingRate(con, building);
7  handleHeatingRate(con, building);
8
9  writeOutputObjs();
```

Figure 6.16.: Result handler method process.

Figure 6.17 exemplarily shows the implementation of the private method *handleHeatingRate*. In line 5 a new instance of *ZoneIdealLoadsZoneTotalHeatingRate* is created which executes a SQL statement (see chapter C listing C.4) to query the SQLite file for the EnergyPlus output variable *"Zone Ideal Loads Zone Total Heating*

Rate". The resulting values are passed to *CityGmlEnergyPlusEnergySystemObject-Factory* class in order to create EnergyADE *EnergyDemandType* (line 6 and 7). In line 9 a new instance of *HeatingRateOutput* is created, which encapsulates the previously created *EnergyDemandType*. in line 10 the instance is added to a list for further processing later on.

```
1   private void handleHeatingRate(Connection con,
2       AbstractBuilding building) throws CityGmlException {
3
4    for (ZoneIdealLoadsZoneTotalHeatingRateResult result :
5           new ZoneIdealLoadsZoneTotalHeatingRate(con).getValues()) {
6      EnergyDemandType energyDemand = CityGmlEnergyPlusEnergySystemObjectFactory
7                                      .createEnergyDemandForThermalZone(...);
8
9      HeatingRateOutput output = new HeatingRateOutput(...);
10     heatingObjs.add(output);
11   }
12   [...]
13  }
```

Figure 6.17.: Result handler method handleHeatingRate.

Finally the private method *writeOutputObjs* is called (see the implementation in figure 6.18). This method is responsible for writing the previously acquired result values back to the CityGML document utilizing the *EnergySystemWriter* class (line 3, 8, 13, 18 and 23).

```
1   private void writeOutputObjs() throws CityGmlWriterException, CityGmlException {
2       for (HeatingRateOutput heatingOutput : heatingObjs) {
3           energySystemWriter.writeEnergyDemandToThermalzone(heatingOutput.getThermalZone(),
4                   heatingOutput.getEnergyDemand());
5       }
6
7       if (heatingCombined != null) {
8           energySystemWriter.writeEnergyDemandToBuilding(heatingCombined.getBuilding(),
9                   heatingCombined.getEnergyDemand(), EndUseType.SPACE_HEATING);
10      }
11
12      for (CoolingRateOutput coolingOutput : coolingObjs) {
13          energySystemWriter.writeEnergyDemandToThermalzone(coolingOutput.getThermalZone(),
14                  coolingOutput.getEnergyDemand());
15      }
16
17      if (coolingCombined != null) {
18          energySystemWriter.writeEnergyDemandToBuilding(coolingCombined.getBuilding(),
19                  coolingCombined.getEnergyDemand(), EndUseType.SPACE_COOLING);
20      }
21
22      for (SolarRadiationHeatGainOutput solarOutput : solarHeatGainObjs) {
23          energySystemWriter.writeGlobalSolarIrradianceToBoundarySurface(solarOutput.getBuilding(),
24                  solarOutput.getAbstractBoundarySurfaceId(), solarOutput.getTimeseries());
25      }
26  }
```

Figure 6.18.: Result handler method writeOutputObjs.

Currently the following simulation results are exported to CityGML:

- *EnergyDemand*

  The EnergyDemand object describes the energy required to satisfy a specific end use of a given object, like space heating, space cooling or domestic hot water. Since it inherits from *_CityObject* it may be associated to any given object (see chapter 4.3.4) [url, 2016d]. Currently, energy demands for cooling and heating are written to each *ThermalZone* (see listing 6.3 for an example). Furthermore, combined energy demands for cooling and heating

on whole building level are written to *Building* (see listing 6.4 for an example). The simulated results have a temporal extent of 1 year and a temporal resolution of 1 hour.

```
<energy:ThermalZone gml:id="id1">
  <gml:description>LoD2 whole-building thermal zone</gml:description>
  <gml:name>LoD2 whole-building thermal zone</gml:name>

  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id1_energy_demand_SpaceHeating">
      <energy:endUse>SpaceHeating</energy:endUse>
      <energy:energyAmount>
        <energy:RegularTimeSeries gml:id="
            id1_timeseries_energydemand_SpaceHeating">
          <!-- Specification of the time series temporal extent and values (
              omitted here) -->
        </energy:RegularTimeSeries>
      </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>
</energy:ThermalZone>
```

Listing 6.3: Example - Energy demand thermal zone

```
<bldg:Building gml:id="id1">
<gml:description>This is a single-part building</gml:description>
  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id1_energy_demand_SpaceHeating">
      <energy:endUse>SpaceHeating</energy:endUse>
      <energy:energyAmount>
        <energy:RegularTimeSeries gml:id="
            id1_timeseries_energydemand_SpaceHeating">
          <!-- Specification of the time series temporal extent and values (
              omitted here) -->
        </energy:RegularTimeSeries>
      </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>

  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id1_energy_demand_SpaceCooling">
      <energy:endUse>SpaceCooling</energy:endUse>
      <energy:energyAmount>
        <energy:RegularTimeSeries gml:id="
            id1_timeseries_energydemand_SpaceCooling">
          <!-- Specification of the time series temporal extent and values (
              omitted here) -->
        </energy:RegularTimeSeries>
      </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>
</bldg>
```

Listing 6.4: Energy demand for whole building.

- *GlobalSolarIrradiance*

_BoundarySurface attribute *GlobalSolarIrradiance* is described as "the sum of the direct, diffuse and reflected irradiance incident on a outside boundary surface and is generally expressed in Watts per square metre." [url, 2016d] and can be related to any outside *_BoundarySurface* (see chapter 4.3.1). The simulated results have a temporal extent of 1 year and a temporal resolution of 1 hour. See listing 6.5 for an example.

```
<bldg:RoofSurface gml:id="id1">
  <gml:description>This is a roof surface</gml:description>
  <gml:name>RoofSurface</gml:name>
  <energy:globalSolarIrradiance>
    <energy:RegularTimeSeries gml:id="id1_timeseries">
      <!-- Specification of the time series temporal extent and values (
          omitted here) -->
    </energy:RegularTimeSeries>
  </energy:globalSolarIrradiance>
</bldg:RoofSurface>
```

Listing 6.5: Global solar irradiance on boundary surface.

6.6. Constraints and prerequisites

During the conception and implementation of the bidirectional data interface between CityGML and EnergyPlus a series of constraints came up. The proposed solutions will be discussed in this chapter.

Up until CityGML Energy ADE v0.5 the definition of thermal hull geometries of thermal zones was only possible through CityGML *_BoundarySurface* classes (see chapter 4.3.1). When working with single thermal zone buildings, this poses no problem, as the *_BoundarySurface* geometries equal the thermal hull geometries. Problems arise when working with multi zone buildings, as EnergyPlus needs precise geometries for each thermal zone (see figure 6.19). Since Energy ADE v0.6 the *ThermalBoundary* class got extended with a additional attribute named *surfaceGeometry* of type *gml:Multisurface*, which provided for flexibility considering multi zone buildings.

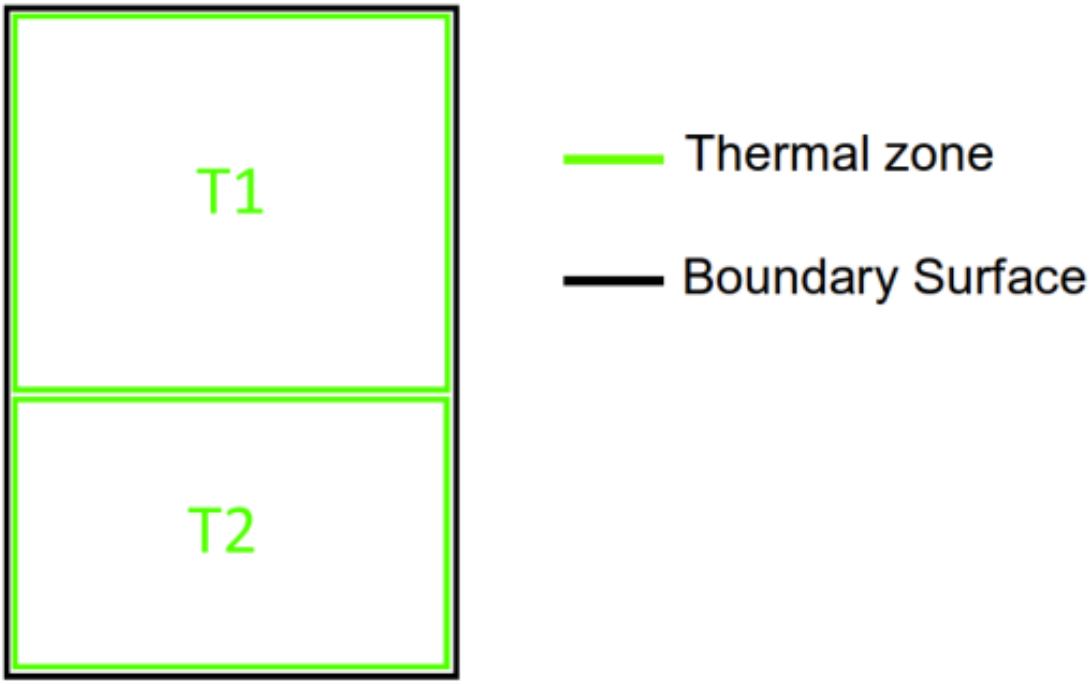

Figure 6.19.: Constraint thermal hull.

Another point to be considered is that Energy Plus expects thermal zones to be convex if a simulation is required with exterior and interior solar irradiance, where EnergyPlus calculates the amount of beam radiation falling on each surface in the zone by projecting the suns rays through the exterior windows. If any straight line passing trough the zone at most intercepts to surfaces, the zone is considered to be convex [Dep, 2015c]. Figure 6.20 shows examples of convex and non-convex zones.

Regarding openings, such as windows and doors, certain prerequisites are required (see figure 6.21) in order to avoid errors during EnergyPlus simulation runs. Openings in EnergyPlus are modeled as subsurfaces with their own geometry, meaning

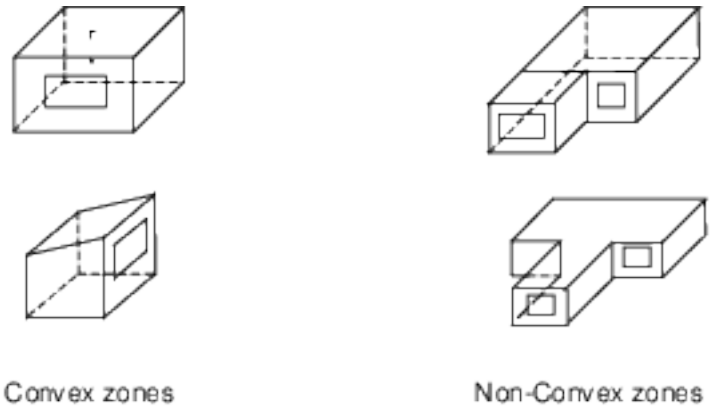

Figure 6.20.: Illustration of Convex and Non-convex Zones. Source: [Dep, 2015c]

the parent surface may not contain holes. In CityGML on the other hand, the geometry of a surface must have hole, which is then filled by the geometry of a opening. Furthermore openings must relate to walls. They reference their parent surface by id (see chapter 6.4). Additionally openings must be of regular (rectangular) shape, they must not touch each other (see figure 6.21 point 1) and they can share at most 1 edge with their parent surface.

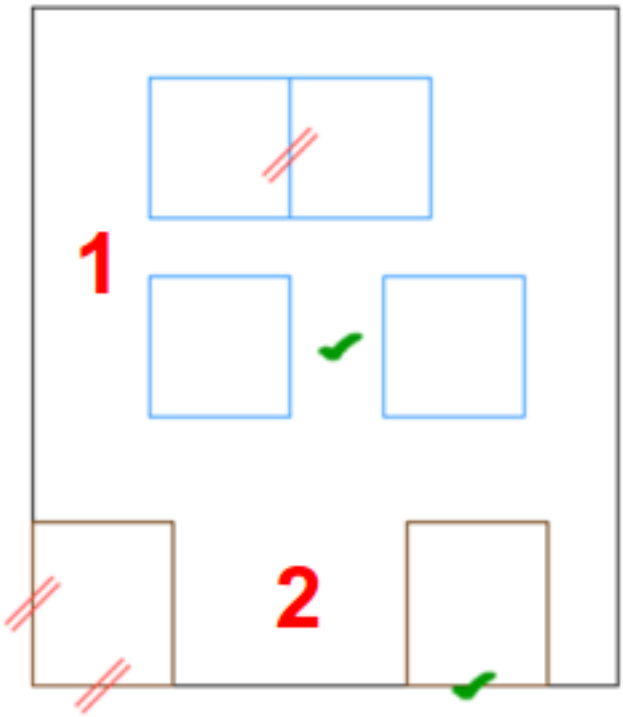

Figure 6.21.: Constraints regarding openings.

In CityGML a thermal zone may relate to 0 or more usage zones (see chapter 4). EnergyPlus does not support this differentiation. With multiple usage zones modeled, preprocessing would mean to "aggregate" them to one usage zone.

Furthermore EnergyPlus needs to distinguish between adjacent ceiling and floors. As of Energy ADE v0.6 *ThermalBoundaryTypeValues* enumeration in the *BuildingPhysics* module (see chapter 4.3.1) supports the following distinctions: *InteriorWall*, *IntermediaryFloor*, *SharedWall*, *OuterWall*, *BasementFloor*, *BasementCeiling*, *AtticFloor* and *Roof*. As a proposed temporary solution a new enumeration value *IntermediaryCeiling* was introduced.

A minor constraint, but nonetheless one to be considered is that EnergyPlus only allows for up to 10 layers when modeling constructions. Energy ADE has no limitations in that regard.

In order to cope with the constraints met, the CityGML input data needs to be preprocessed, which is not topic of this thesis and considered as given.

7. Experimental results

7.1. Introduction

After introducing and describing the concepts and implementation details of the bidirectional data interface between CityGML and EnergyPlus a translation between the both of them can be performed. This chapter provides an overview of the testing of the bidirectional data interface and the evaluation of the results. Section 7.2 gives an overview of the test data used, followed by the discussion of the results in section 7.3. Section **??** shows ways of integrating the results into a GIS.

7.2. Test data description

This section provides an introduction of the data used during the conception and implementation of the bi-directional data interface. In section 7.2.1 the focus lies on a CityGML 2.0 data set containing a single building modeled in LoD2 and LoD3 enriched with several Energy ADE v0.6 objects including thermal zones, usage zones, occupants, facilities, daily schedules, constructions and materials and energy demands. This CityGML document was constructed from the ground up and served as the main test data during implementation of the data interface, that got adopted whenever constraints or limitations where met. The data set talked about in section 7.2.2 on the other hand consists of 46 real-world LoD2 buildings situated in the 12th district of Vienna/Austria.

The test data has been pre-processed in order to meet the constraints discussed in chapter 6.6.

7.2.1. Single building with multiple zones

Figure 7.1 shows a graphical representation of the building developed from the ground up at the Austrian Institute of Technology[1]. In terms of geometry, the building is modeled in LoD2 and LoD3 providing semantically differentiation according to CityGML (see chapter 3). The building consists of 5 different convex thermal zones and 5 different usage zones. The usage zones are categorized as followed: 2 residential, 1 commercial and 1 school. Each usage zone provides information about

[1] http://www.ait.ac.at/

domestic hot water facilities, electrical appliances, lighting facilities, as well as their operation schedules.

- Ground surface area, volume, number of storeys
- Heating energy demand for the building and each thermal zone (monthly values)
- Year of construction
- Refurbishment measures on the building and energy performance certificates
- Energy Performance Certification

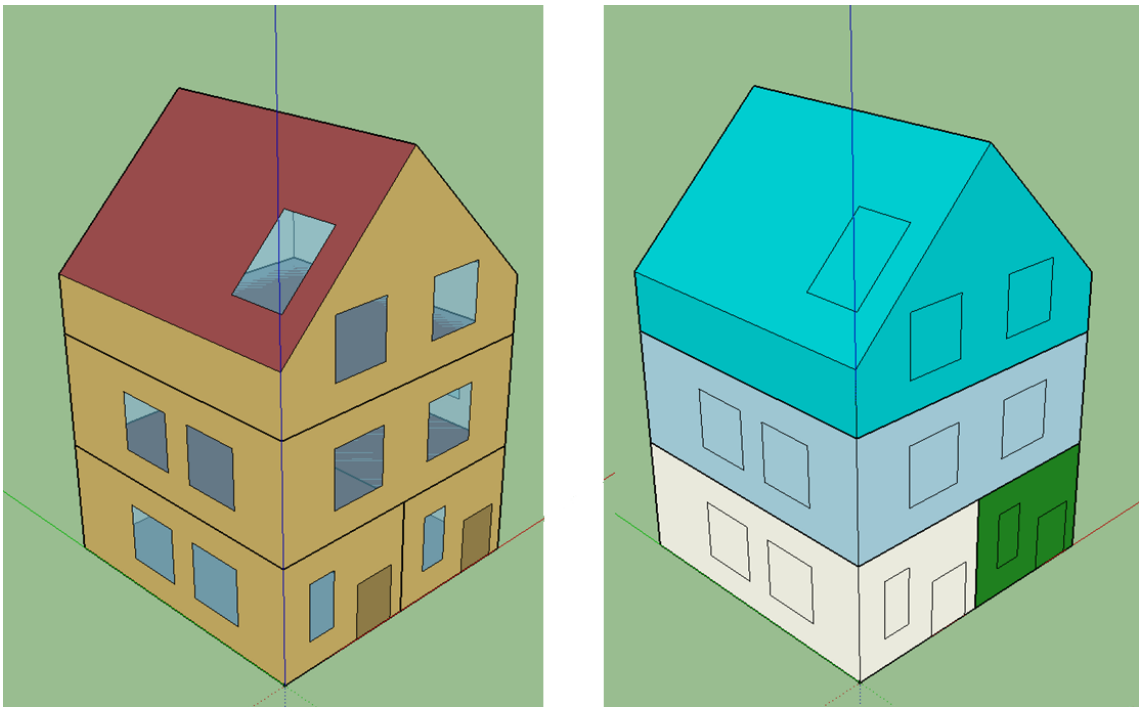

Figure 7.1.: Building colored by surface type (left) and thermal zones (right).

7.2.2. Multiple buildings with a single zone

The test data talked about in this section consists of 46 real-world buildings situated in Meidling, the 12th district of Vienna/Austria with the area being about 300x250 m wide, extracted from the integrated CityGML-based 3D model of Vienna described by Agugiaro in [Agugiaro, 2016b]. The 3D city model was obtained by integrating numerous heterogeneous data sources, which were either readily available trough the Open Government Data[2] initiative by the City of Vienna or made available trough the "CINERGY[3], Smart cities with sustainable energy systems" project [Agugiaro and Möller] (see figure 7.2). In terms of geometry each building is

[2]https://open.wien.gv.at/site/open-data/
[3]http://ci-nergy.eu/

available in LoD2 providing semantically differentiation according to CityGML (see chapter 3).

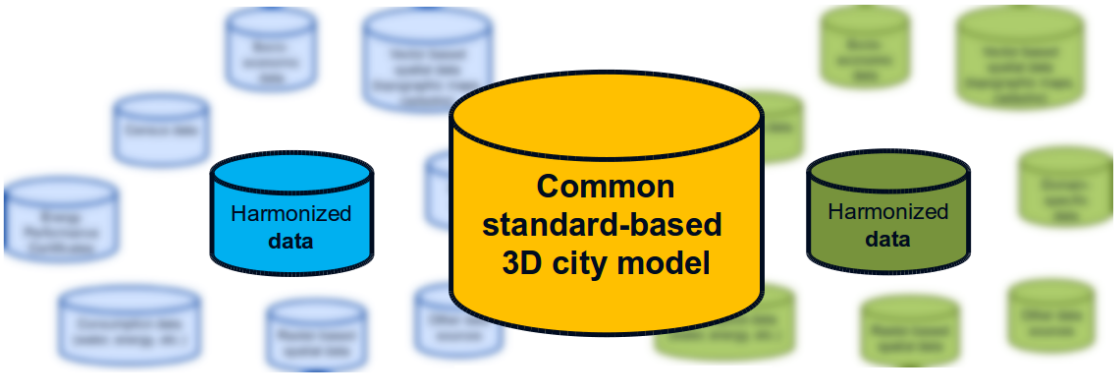

Figure 7.2.: Integrated common 3D city model combining different harmonized data.

All spatial data are geo-referenced according to MGI/Austria GK East projection with EPSG code 31256. Spatial data sources integrated into the test data include: *Mehrzweckkarte* (MZK) of Vienna, *Baukörpermodell* containing a prismatic representation of buildings, CityGML files containing LoD2 buildings modelled from Lidar and photogrammetric DTM data, a vector map containing the land-use, several point-based vector maps providing information about buildings (like building names, use/function) and vector-based maps prepresenting administrative boundaries. Non-spatial data sources integrated into the test data include: *Wiener Wärme Kataster* (WWK) providing a series of attributes (like address, building block id, year of construction, and so on), an XML-based file containing information about the social housing buildings of Vienna and data about number of households, their surface net area and so on for a limited number of buildings [Agugiaro, 2016b].

Each building consists of 1 convex thermal zone and 1 usage zone. Each usage zone provides information about domestic hot water facilities, electrical appliances, lighting facilities, as well as their operation schedules.

Each building additionally integrates the following data:

- Name and address
- Class, function and usage of the building
- Year of construction
- Number of storeys
- Measured height
- Gross volume
- Gross and net floor area
- Roof type

Figure 7.3 and 7.4 show a graphical representation of the test area.

Figure 7.3.: 46 buildings located in Meidling (Vienna/Austria).

Figure 7.4.: 46 buildings located in Meidling (Vienna/Austria).

7.3. Results

This section discusses the results generating by test runs of the bi-directional data interface. As already mentionend, one of the goals of this thesis was to exploit the benefits of a dynamic energy simulation software, namely EnergyPlus coupled with CityGML and its Energy ADE in order to automatize energy simulations of buildings at urban scale.

As already mentioned in chapter 5 EnergyPlus supports the input of real weather data during simulation runs. The weather data for Vienna used during the test runs can freely be downloaded from the EnergyPlus weather data site[4].

During the urban scale simulation run regarding the test area of Meidling in Vienna/Austria, no shadowing of neighbouring buildings has been taken into account.

Once the execution pipeline of data interface gets started, each building in the input CityGML document is completely translated into EnergyPlus Input Data File (IDF) format, followed by the actual simulation itself. The following output values were considered: hourly heat gain rate of incident solar radiation on the surfaces, hourly energy demand for cooling and heating for each zone, combined hourly energy demands for cooling and heating on building level. These values are subsequently extracted and written back to the CityGML document.

Figure 7.5 exemplarily shows the hourly results of a heating demand of a thermal zone and the heat gain rate of incident solar radiation of a surface. These plots were manually created from the EnergyPlus result files.

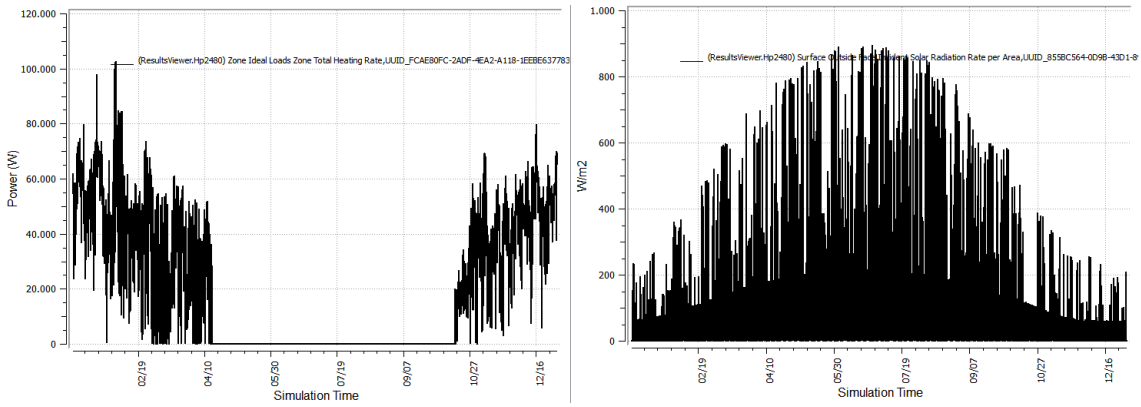

Figure 7.5.: Plot of heating demand (left) and heat gain rate of incident solar radiation on a surface (right).

The CityGML input document is enriched with yearly energy demand time series with a temporal resolution of 1 hour for heating and cooling demands per thermal zone and on building level. The following listings are direct excerpts from the resulting CityGML document. Listing 7.1 and 7.2 show examples of heating and cooling energy demands for thermal zones and buildings, respectively.

[4]https://energyplus.net/weather

```
<energy:ThermalZone gml:id="id1">
  <gml:description>LoD2 whole-building thermal zone</gml:description>
  <gml:name>LoD2 whole-building thermal zone</gml:name>

  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id_energy_demand_SpaceCooling_UUID_fcae80fc-2adf
        -4ea2-a118-1eebe6377830">
    <energy:endUse>SpaceHeating</energy:endUse>
    <energy:energyAmount>
      <energy:RegularTimeSeries gml:id="UUID_fcae80fc-2adf-4ea2-a118-1
          eebe6377830_timeseries_energydemand_SpaceCooling">
        <energy:timeInterval unit="hour">1</energy:timeInterval>
        <!-- values omitted here -->
      </energy:RegularTimeSeries>
      </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>

  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id_energy_demand_SpaceHeating_UUID_fcae80fc-2adf
        -4ea2-a118-1eebe6377830">
    <energy:endUse>SpaceHeating</energy:endUse>
    <energy:energyAmount>
      <energy:RegularTimeSeries gml:id="UUID_fcae80fc-2adf-4ea2-a118-1
          eebe6377830_timeseries_energydemand_SpaceHeating">
        <energy:timeInterval unit="hour">1</energy:timeInterval>
        <!-- values omitted here -->
      </energy:RegularTimeSeries>
    </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>
</energy:ThermalZone>
```

Listing 7.1: Heating and cooling energy demands for a thermal zone.

```
<bldg:Building gml:id="UUID_fe756221-1681-422a-8935-2d302e696bea">
  <gml:description>This is a single-part building</gml:description>
  <creationDate>2016-04-07</creationDate>
  <energy:energyDemands>
    <energy:EnergyDemand gml:id="id_energy_demand_SpaceHeating_UUID_fe756221
        -1681-422a-8935-2d302e696bea">
      <gml:description/>
      <gml:name/>
      <energy:endUse>SpaceHeating</energy:endUse>
      <energy:energyAmount>
        <energy:RegularTimeSeries gml:id="UUID_fe756221-1681-422a-8935-2
            d302e696bea_timeseries_energydemand_SpaceHeating">
          <energy:timeInterval unit="hour">1</energy:timeInterval>
          <!-- values omitted here -->
        </energy:RegularTimeSeries>
      </energy:energyAmount>
    </energy:EnergyDemand>
  </energy:energyDemands>
  <energy:energyDemands>
```

```
18      <energy:EnergyDemand gml:id="id_energy_demand_SpaceCooling_UUID_fe756221
            -1681-422a-8935-2d302e696bea">
19        <gml:description/>
20        <gml:name/>
21        <energy:endUse>SpaceCooling</energy:endUse>
22        <energy:energyAmount>
23          <energy:RegularTimeSeries gml:id="UUID_fe756221-1681-422a-8935-2
                d302e696bea_timeseries_energydemand_SpaceCooling">
24            <energy:timeInterval unit="hour">1</energy:timeInterval>
25            <!-- values omitted here -->
26          </energy:RegularTimeSeries>
27        </energy:energyAmount>
28      </energy:EnergyDemand>
29    </energy:energyDemands>
30    <bldg:class>1000</bldg:class>
31    <bldg:function>1000</bldg:function>
32    <bldg:usage>1000</bldg:usage>
33    <bldg:yearOfConstruction>1970</bldg:yearOfConstruction>
34    <bldg:roofType>Satteldach</bldg:roofType>
35    <bldg:measuredHeight uom="m">12.72</bldg:measuredHeight>
36    <bldg:storeysAboveGround>3</bldg:storeysAboveGround>
37    <bldg:storeysBelowGround>0</bldg:storeysBelowGround>
38  </bldg:Building>
```

Listing 7.2: Combined heating and cooling energy demands for a building.

Listing 7.3 shows an example of a global solar irradiance for a wall surface.

```
1  <bldg:WallSurface gml:id="UUID_LOD2_205728-1cd45609-47cb-4046-a0b6_4dc2c404
       -4106-495b-bbfd-5be289c9165e">
2    <gml:description>This is a wall surface</gml:description>
3    <gml:name>WallSurface</gml:name>
4    <creationDate>2016-04-07</creationDate>
5    <energy:globalSolarIrradiance>
6      <energy:RegularTimeSeries gml:id="UUID_LOD2_205728-1cd45609-47cb-4046-
           a0b6_4dc2c404-4106-495b-bbfd-5be289c9165e_timeseries">
7        <energy:timeInterval unit="hour">1</energy:timeInterval>
8        <!-- values omitted here -->
9      </energy:RegularTimeSeries>
10   </energy:globalSolarIrradiance>
11   <bldg:lod2MultiSurface>
12   ...
13   </bldg:lod2MultiSurface>
14 </bldg:WallSurface>
```

Listing 7.3: Global solar irradiance on a surface.

The test results show, that harmonizing different heterogeneous data sources of spatial and non-spatial nature into a common integrated information hub allow for a wide range of applications. Exploiting such a data model to run energy demand simulations and store the results back to the data model to serve as a basis for urban planning processes, as presented in this thesis, being just one of them. It has been shown, that problems, such as difficulties of obtaining city-wide information because of often inaccurate, missing or not properly integrated data spread out over different heterogeneous data sources [Agugiaro et al., 2015], can be solved.

The result data can be inserted into a CityGML/ADE compliant database to be used for different applications. The 3D City Database[5] (3DCityDB) is a free Open Source package providing a database schema (available for PostgreSQL[6]/PostGis[7] and Oracle Database[8]) and a number of tools to store, represent and manage CityGML models. The package also contains a graphical user interface (see figure 7.6) providing functionalities for the direct export of CityGML models in KML, COLLADA and glTF formats, which allows to visualize 3D city models in GIS applications or digital virtual globes like Google Earth or Cesium[9] [Kolbe et al., 2016]. 3DCityDb is being developed mainly by the Chair of Geoinformatics at the Technische Universität München.

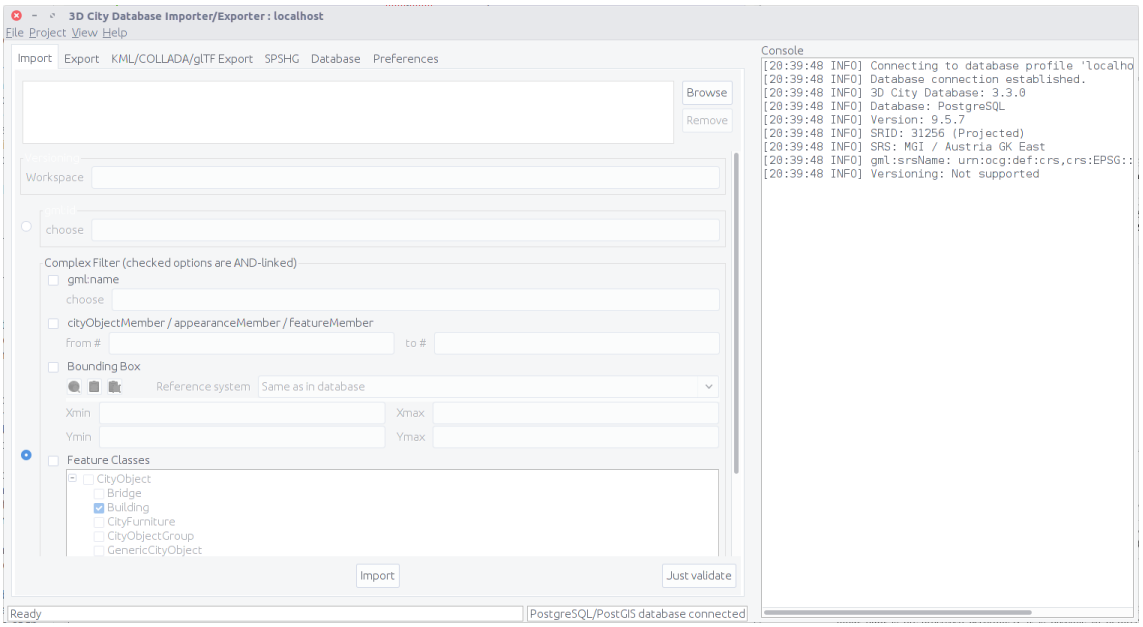

Figure 7.6.: 3DCityDB Importer/Exporter.

The next step is to export features using the KML/COLLADA/glTF Export screen (see figure 7.7) to create KML/KMZ files usable by Google Earth. Figure 7.8 shows an example of the exported CityGml model exported in KMZ and visualized in Google Earth.

The user can click on the corresponding geometry and retrieve information, which is presented directly inside a pop-up balloon. The user is also able to edit all meaningful parameters manually and to invoke computations, which directly recompute all results immediately and on the fly [Agugiaro, 2015]. Figure 7.9 shows the detailed information of heating energy demand of the selected building providing information about monthly heating energy demand (left) and using finer-grained temporal resolution of hourly values provided by the data interface described in this thesis (right) after clicking on a building's geometry.

[5] `http://www.3dcitydb.org/3dcitydb/3dcitydbhomepage/`
[6] https://www.postgresql.org/
[7] `http://postgis.net/`
[8] `https://www.oracle.com/database/index.html`
[9] `https://cesiumjs.org/`

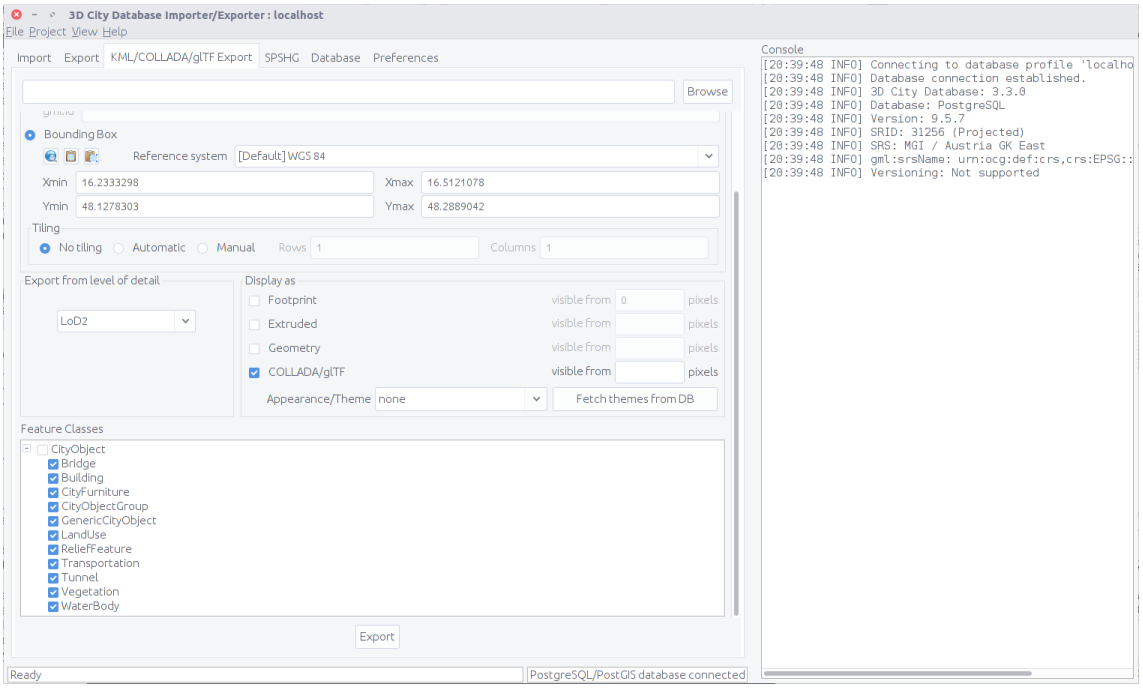

Figure 7.7.: 3DCityDB Importer/Exporter KML/COLLADA/glTF Export Screen.

Figure 7.8.: Exported CityGML features visualized in Google Earth.

Starting with release 3.3.0 the 3DCityDB software package includes a Cesium-based "3DCityDB-Web-Map-Client", which acts as a web-based front-end for interactive visualization of 3D City models [Kolbe et al., 2016]. Cesium[10] is an open-source

[10] `https://cesiumjs.org/`

Figure 7.9.: 3D city model of Meidling in Vienna/Austria with monthly heating energy demand (left) and hourly heating energy demand (right).

JavaScript library for 3D globes and maps developed by Analytical Graphics, Inc., which provides cross-platform functionalities on the Web. The 3D web client has been functionally extended, allowing for example to directly visualize exported KML/glTF models via the 3DCityDB Importer/Exporter. As already mentioned in chapter 2 existing 3D model used for visualization lack in terms of the ability to handle semantic information. The 3D web client therefore supports linking the models with table data exported using the Spreadsheed Generator Plugin and uploaded to an online spreadsheet (namely Google Fusion Table[11]) to query thematic data [Kolbe et al., 2016]. Figure 7.10 shows an example of thematic data on Google Fusion Tables linked to the 3D web client.

The 3D web client streamlines the process of providing ways to visualize and analyze the integrated information hub, that is CityGML. First tests have been implemented at the Austrian Institute of Technology (AIT) in Vienna to visualize energy scenario information in the 3D web client (see figure 7.11 and 7.12). Altough, at the time of writing the 3DCityDB does not support handling CityGML extensions in the form of ADEs (it is being worked on). Right now, the specific EnergyADE elements would have to be transformed into generic attributes, which subsequently could be exported.

[11] `https://www.google.com/fusiontables`

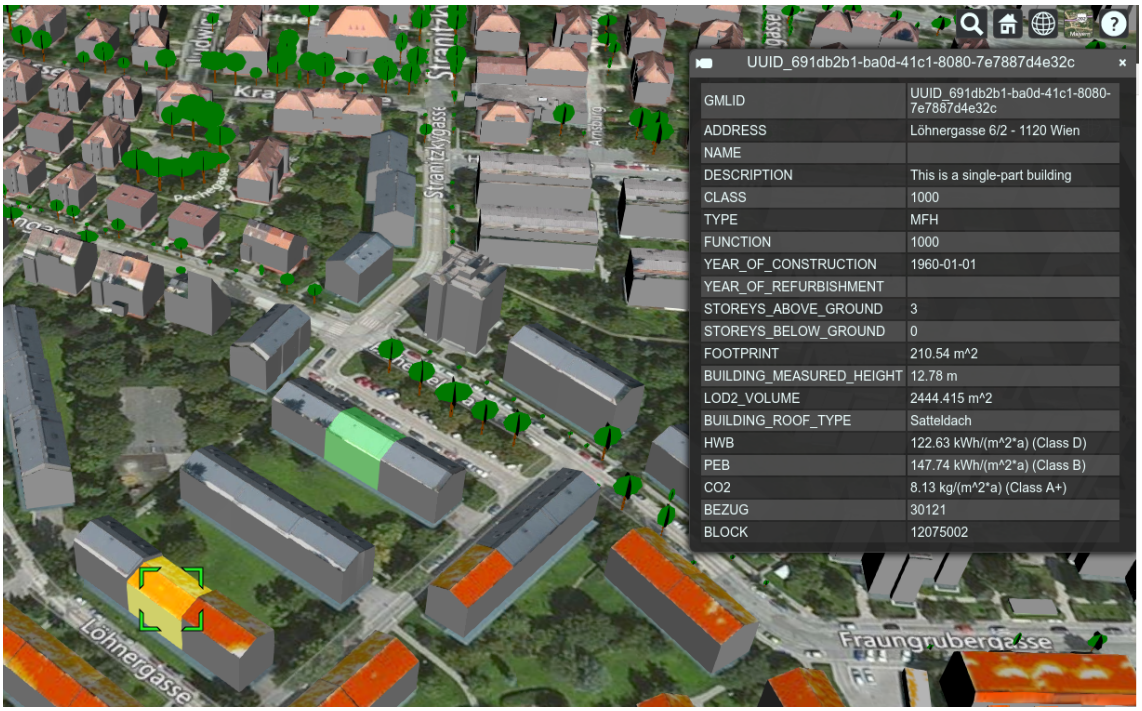

Figure 7.10.: Example of thematic data on Google Fusion Tables linked to 3DCityDB-Web-Map-Client.

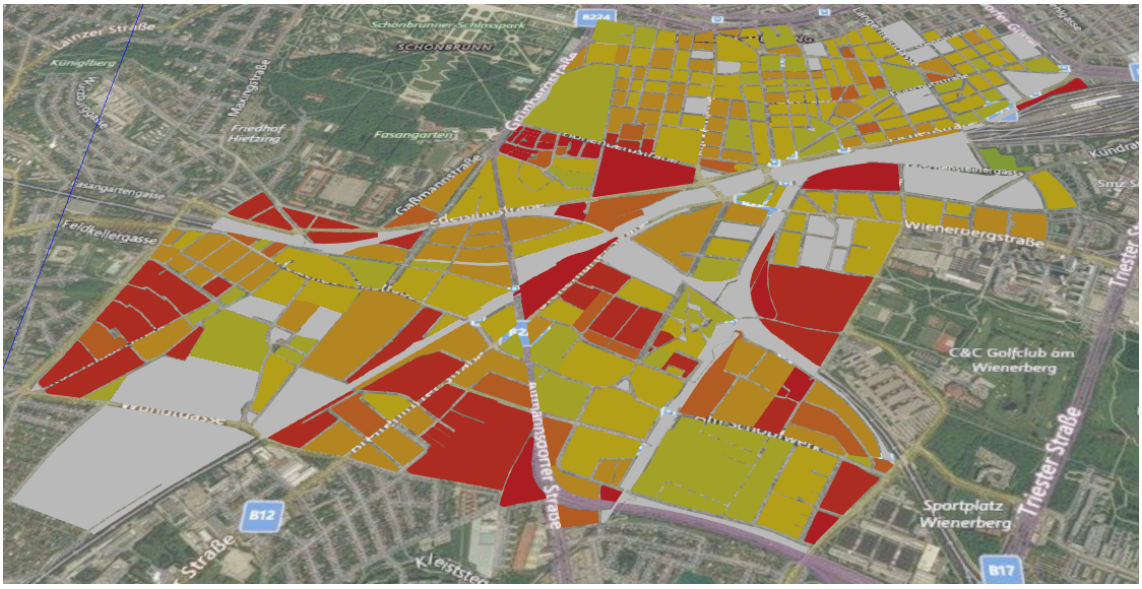

Figure 7.11.: Visualizing energy scenario information in 3DCityDB-Web-Map-Client - Overview.

Figure 7.12.: Visualizing energy scenario information in 3DCityDB-Web-Map-Client - Showing feature attributes.

8. Contributions to the Energy ADE development

As already discussed in section 6.6 a series of constraints came up during the conception and implementation of the bidirectional data interface between CityGML and EnergyPlus.

This chapter will discuss the contributions made to the development of the Energy ADE.

Up until CityGML Energy ADE v0.5 the definition of thermal hull geometries of thermal zones was only possible through CityGML *_BoundarySurface* classes. As noted in section 6.6, this poses a problem, as soon as multi zone buildings are involved, as EnergyPlus needs precise geometries for each thermal zone. This problem was discussed at the semiannual Energy ADE workshop which took place in Munich in December 2015[1]. The proposed solution included an introduction of additional optional geometric attributes to the *ThermalBoundarySurface* class. See figure 8.1 showing *ThermalZone* and *ThermalBoundarySurface* classes of the Building Physics module in the Energy ADE v0.5 and the agreed solution as represented in Energy ADE v0.8 in figure 8.2.

Furthermore, as stated in chapter 6.6 EnergyPlus needs to distinguish between adjacent ceiling and floors. As of Energy ADE v0.6 *ThermalBoundaryTypeValues* enumeration in the *BuildingPhysics* module (see chapter 4.3.1) supports the following distinctions: *InteriorWall*, *IntermediaryFloor*, *SharedWall*, *OuterWall*, *BasementFloor*, *BasementCeiling*, *AtticFloor* and *Roof*. As a proposed temporary solution a new enumeration value *IntermediaryCeiling* was introduced (see figure 8.3). However, after discussions made at the semiannual Energy ADE workshop in Vienna in May 2016[2] it was agreed on, that the required information should be included in the definition of the ordered relation between ThermalZone and ThermalBoundarySurface (see relation *delimitsBy* in figure 8.2).

The proposition recap is as follows (see https://github.com/cstb/citygml-energy/issues/109):

- The first *Construction Layer* is facing outside, the last *Construction Layer* is facing inside (or contrary if *ConstructionOrientation* is false).
- The last layer is facing the first *ThermalZone* and the first layer is facing the second *ThermalZone* of the relation "delimitsBy".

[1] http://en.wiki.energy.sig3d.org/index.php/Workshop_Munich_2015
[2] http://en.wiki.energy.sig3d.org/index.php/Workshop_Vienna_2016

- For an intermediary floor, the first layer faces the *ThermalZone* below, while the last layer face the *ThermalZone* above.

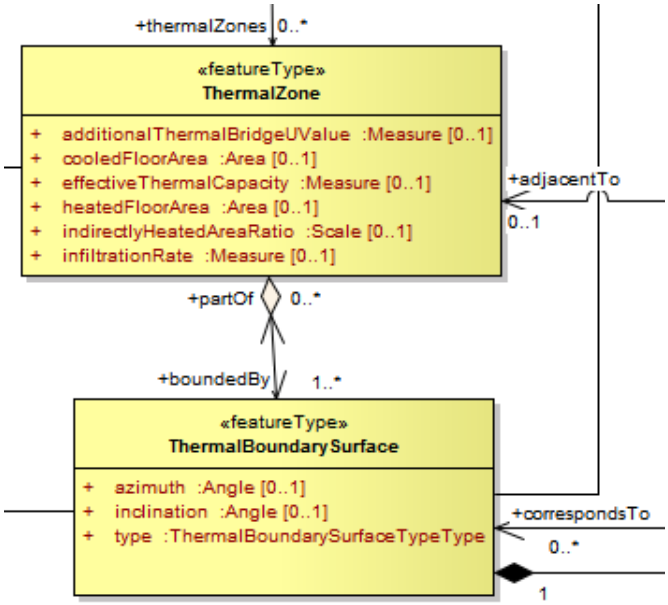

Figure 8.1.: UML diagram of the Building Physics module in the Energy ADE v0.5 showing ThermalZone and ThermalBoundarySurface. Source: [url, 2016d]

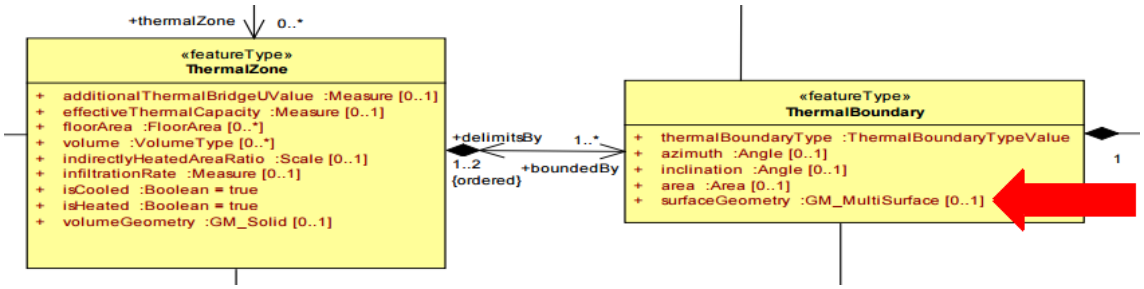

Figure 8.2.: UML diagram of the Building Physics module in the Energy ADE v0.8 showing ThermalZone and ThermalBoundarySurface. Source: [url, 2016d]

Figure 8.3.: UML diagram of the Building Physics module in the Energy ADE v0.5 (left) and v0.8 showing ThermalZone and ThermalBoundarySurface. Source: [url, 2016d]

9. Summary

In this master thesis the coupling of a CityGML-based semantic city model with energy simulation software, in order to profit from a dynamic energy simulation resulting in fine-grained energy performance data on an urban scale providing additional value for different entities from different domains was presented and described.

CityGML is an XML-based open data model used for the storage, manipulation, presentation and data exchange of semantic virtual 3D city models at urban and territorial scale. It defines a model describing not only geometry, but also topology, semantics and appearance with 5 possible (geometric and semantic) levels of detail (LoD). CityGML provides a mechanism called Application Domain Extension (ADE) allowing other domains to specify additions to the existing data model.

An example of such extension is the Energy ADE which allows for the representation, storage and exchange of energy-related features and attributes relevant to urban energy models. Like its parent CityGML the Energy ADE is structured to be modular, for the sake of extendibility or reusability.

Next up the energy simulation program EnergyPlus was discussed. EnergyPlus is a whole building energy simulation program used to model energy and water use in buildings. It allows for dynamic sub-hourly (different from the fixed time steps used by most programs) energy performance simulation incorporating real weather data.

This thesis should answer question *"Is it possible to incorporate standardized, spatio-semantic coherent 3D city model (CityGML) into already existing dynamic software simulation tools (EnergyPlus) to create precise hourly energy performance data at urban scale?"* or more generally *"to exploit a standardized, spatio-semantic coherent 3D city model to be used as basis for different kind of applications"* to be visualized and communicated to end-users of different areas. In order to answer the question a bi-directional data interface between CityGML and EnergyPlus was conceptually modeled and implemented. Conception involved finding the relevant EnergyPlus classes needed to perform an energy simulation and mapping them to the equivalent CityGML/Energy ADE counterparts. The data interface converts CityGML data to EnergyPlus Input Data Files (IDF), launches EnergyPlus simulations, extracts relevant simulation results and integrates them back into CityGML. The data interface is based on the Energy ADE v0.6 (Conception started at Energy ADE v0.5 though). In the course of conception and implementation a series of constraints were met. Up until Energy ADE v0.5 the definition of the thermal hull geometries was only possible trough *_BoundarySurface* class, which would have been problematic working with buildings consisting of multiple thermal zones. Trough active input to

the Energy ADE consortium the next release of Energy ADE v0.6 removed this constraint by introducing additional properties to the *ThermalBoundary* class allowing for more flexibility. Also in order to fully exploit EnergyPlus capabilities like exterior and interior solar irradiance calculations the thermal zones need to be convex. Also openings, such as windows and doors require special handling. EnergyPlus expects them to be subsurfaces of a parent surface (for example a wall) with their own geometry, meaning the parent can not have any holes. CityGML on the other hand allows for holes in the geometries. Openings must further be of rectangular shape and the must not touch each other and can share at most 1 edge with their parent. EnergyPlus furthermore has no distinction between thermal zone and usage zone, unlike CityGML with the Energy ADE. In order to work with multiple usage zone, one must aggregate them to one. EnergyPlus needs to distinguish between adjacent ceiling and floors. Since Energy ADE only provides *IntermediaryFloor* as thermal boundary type the schema had to be adjusted. A change request for the next version of the Energy ADE has been submitted in that regard. However, after discussions in the semiannual Energy ADE workshop in Vienna in May 2016 another solution was proposed as discussed in chapter 8. To test the data interface 2 data sets have been defined. One being a single building with multiple zones with LoD2 and LoD3 geometries built from the ground up and one being real-world data consisting of 46 LoD2 buildings with a single zone situated in Meidling in Vienna/Austria. The results have further been integrated into Google Earth and experimentally into 3DCityDB-Web-Map-Client as tools of communication to for end-users of different areas. The thesis proved the test to be successful, so the question asked beforehand can be answered provided that the CityGML input data is pre-processed according to the constraints met during the conception and implementation of the data interface.

10. Outlook and future improvements

The last section of this thesis addresses open questions and research areas to be considered in future developments.

Regarding future improvements of the data interface the following suggestions come up. EnergyPlus does not support simulations of more than one building. In case of urban scale energy performance simulations that include solar radiation, shadowing of other buildings is a point to be considered, as it will most likely affect the outcome of the simulation. All external surfaces in buildings other than the current one could be modelled as shading surfaces. Modeling adjacency to other buildings could be achieved by using the adiabatic adjacency option in the touching surface.

Since only a fraction of the available Energy Plus classes available are currently used, future implementations could implement more mappings of interest between CityGML and EnergyPlus to extend simulation options.

Currently available input and output options are limited to CityGML documents. In the future, directly linking the data interface to CityGML and ADE compliant databases (for example the 3D City DB[1]) would be greatly improve the field of application.

Another point to consider in future implementations is the further integration of the Energy ADE module *Energy System* and an implementation of a graphical user interface to ease the handling of the data interface.

And of course further testing with real data (eg. from Vienna city model) should be seen as a constant work in progress.

[1] http://www.3dcitydb.org/3dcitydb/3dcitydbhomepage/

Bibliography

Ade-xjc - xml schema binding compiler for citygml ades. https://github.com/citygml4j/ade-xjc, 2016. Accessed: 01.07.2016.

Citygml4j - the open source java api for citygml. https://github.com/citygml4j/citygml4j, 2016. Accessed: 01.07.2016.

Java platform, standard edition 7 api specification. https://docs.oracle.com/javase/7/docs/api/, 2016. Accessed: 01.07.2016.

Cinergy, smart cities with sustainable energy systems. http://www.ci-nergy.eu/index.html, 2016a. Accessed: 01.07.2016.

Collada - 3d asset exchange schema. https://www.khronos.org/collada/, 2016b. Accessed: 01.07.2016.

Citygml energy ade. https://github.com/cstb/citygml-energy, 2016c. Accessed: 01.07.2016.

Citygml energy ade guidelines. https://github.com/cstb/citygml-energy/blob/master/doc/guidelines/Guidelines_EnergyADE.md, 2016d. Accessed: 01.07.2016.

gbxml - green building xml schema. http://www.gbxml.org/About_GreenBuildingXML_gbXML, 2016e. Accessed: 01.07.2016.

gbxml-schema - green building xml schema. https://github.com/GreenBuildingXML/gbXML-Schema, 2016f. Accessed: 01.07.2016.

Ifc - industry foundation classes. http://www.buildingsmart-tech.org/specifications, 2016g. Accessed: 01.07.2016.

G Agugiaro. From sub-optimal datasets to a citygml-compliant 3d city model: experiences from trento, italy. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(4):7, 2014.

G Agugiaro. Enabling energy-awareness in the semantic 3d city model of vienna. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 81–88, 2016a.

G Agugiaro. First steps towards an integrated citygml-based 3d model of vienna. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. III-4*, pages 139–146, 2016b.

Giorgio Agugiaro. Energy planning tools and citygml-based 3d virtual city models: Experiences from trento (italy). *Applied Geomatics*, pages 1–16, 2015.

Giorgio Agugiaro and Sebastian Möller. Gebäudescharfe schätzung des heizwärmebedarfs anhand des semantischen 3d-stadtmodells wiens.

Giorgio Agugiaro, Stefan Hauer, and Florian Nadler. Coupling of citygml-based semantic city models with energy simulation tools: some experiences. *Proceedings REAL CORP 2015*, 2015.

Angela Altmaier and Thomas H Kolbe. Applications and solutions for interoperable 3d geo-visualization. In *Proceedings of the photogrammetric week*, pages 251–267, 2003.

Klement Aringer, Andreas Donaubauer, Thomas H Kolbe, and Robert Roschlaub. Modellbasierte transformation von 3d-gebäudemodellen nach inspire.

J-M Bahu, A Koch, E Kremers, and SM Murshed. Towards a 3d spatial urban energy modelling approach. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1(1):33–41, 2013.

Daniel Banfi. Energiebedarfsanalyse urbaner räume anhand des semantischen modells und austauschformats citygml. Master's thesis, Technische Universität München, 2013.

Mark Barnes and Ellen Levy Finch. Collada-digital asset schema release 1.5. 0. *Sony Computer Entertainment Inc. Atazadeh, Sam Amirebrahimi, Alireza Jamshidi (7048) 3D-Cadastre, a Multifaceted Challenge*, 2008.

Gabor Bartha and Sandor Kocsis. Standardization of geographic data: The european inspire directive. *European Journal of Geography*, 2(2):79–89, 2011.

A Bhatia. Cooling load calculations and principles. *Continuing Education and Development, Inc. New York*, 2001.

Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015.

Barbara Burger. Erzeugung eines semantischen 3d-stadtmodells der stadt new york auf der basis von open data - dgm, straen und zonierung. Masterarbeit, Technische Universitt Mnchen, Mnchen, Jul 2015.

Paola Caputo, Costa Gaia, and Valentina Zanotto. A methodology for defining electricity demand in energy simulations referred to the italian context. *Energies*, 6(12):6274–6292, 2013.

Stephen J. Chapman. *Introduction to Java*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1999. ISBN 0139194169.

Silvia Coccolo, Dasaraden Mauree, Jérôme Kämpf, and Jean-Louis Scartezzini. Integration of outdoor human comfort in a building energy simulation database using citygml energy ade. In *Expanding Boundaries-Systems Thinking in the Built Environment-Proceedings of the Sustainable Built Environment (SBE) Regional Conference Zurich 2016*, number EPFL-CONF-220658. vdf Hochschulverlag AG ETH Zurich, 2016.

Drury B. Crawley, Linda K. Lawrie, Frederick C. Winkelmann, W. F. Buhl, Y. Joe Huang, Curtis O. Pedersen, Richard K. Strand, Richard J. Liesen, Daniel E. Fisher, Michael J. Witte, and Jason Glazer. Energyplus: Creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4): 319–331, 4 2001. ISSN 0378-7788. doi: 10.1016/S0378-7788(00)00114-6.

Drury B Crawley, Linda K Lawrie, Curtis O Pedersen, Frederick C Winkelmann, Michael J Witte, Richard K Strand, Richard J Liesen, Walter F Buhl, Yu Joe Huang, Robert H Henninger, et al. Energyplus: New, capable, and linked. *Journal of Architectural and Planning Research*, pages 292–302, 2004.

Pierre Dagnely, Tom Ruette, Tom Tourwé, Elena Tsiporkova, and Clara Verhelst. Predicting hourly energy consumption. can you beat an autoregressive model. In *Proceeding of the 24th Annual Machine Learning Conference of Belgium and the Netherlands, Benelearn, Delft, The Netherlands*, volume 19, 2015.

EnergyPlus Documentation - Guide for Interface Developers. Department of Energy, Washington, USA, 2015a.

EnergyPlus Documentation - Getting Started with EnergyPlus. Department of Energy, Washington, USA, 2015b.

EnergyPlus Documentation - Input Output Reference. Department of Energy, Washington, USA, 2015c.

Jielin Dong. *Network Dictionary*. Javvin Technologies Inc., 2007.

James Gosling. *The Java language specification*. Addison-Wesley Professional, 2000.

Gerhard Gröger and Lutz Plümer. Citygml interoperable semantic 3d city models. {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, 71:12 – 33, 2012. ISSN 0924-2716. doi: http://dx.doi.org/10.1016/j.isprsjprs.2012.04.004. URL `http://www.sciencedirect.com/science/article/pii/S0924271612000779`.

Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, and Karl-Heinz Häfele. OGC City Geography Markup Language(CityGML) Encoding Standard. Technical report, Open Geospatial Consortium, April 2012.

Mehreen S Gul and Sandhya Patidar. Understanding the energy consumption and occupancy of a multi-purpose academic building. *Energy and Buildings*, 87:155–165, 2015.

J Herring. The opengis abstract specification, topic 1: Feature geometry (iso 19107 spatial schema), version 5. *OGC document*, (01-101), 2001.

John F Hughes, Andries Van Dam, James D Foley, and Steven K Feiner. *Computer graphics: principles and practice*. Pearson Education, 2014.

Developing Spatial Data Infrastructures. the sdi cookbook. *GSDI/Nebert*, 2004.

TWGBU INSPIRE. Inspire data specification on buildings version 3, release candidate 3–draft technical guidelines, identifier d2. 8. iii. 2_v3. 0rc3, inspire thematic working group buildings, 2013.

AKSHEY JAWA. Development and analysis of a tool for speed up of energyplus through parallelization. Masterarbeit, International Institute of Information Technology Hyderabad, Hyderabad - 500 032, INDIA, June 2016.

R Kaden and TH Kolbe. City-wide total energy demand estimation of buildings using semantic 3d city models and statistical data. *Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2:W1, 2013.

TH Kolbe, G König, C Nagel, and A Stadler. 3dcitydb-documentation. *Institute for Geodesy and Geoinformation Science, Technische Universität Berlin*, 2016.

Thomas H Kolbe. Representing and exchanging 3d city models with citygml. In *3D geo-information sciences*, pages 15–31. Springer, 2009.

A Krger and TH Kolbe. Building analysis for urban energy planning using key indicators on virtual 3d city modelsthe energy atlas of berlin. In *Proceedings of the ISPRS Congress*, 2012.

K Kumar, S Saran, and AS Kumar. Citygml based interoperability for the transformation of 3d data models. *Trans. GIS*, 2016.

Azadeh Mazaheri. Modeling of energy storage systems for building intergation. 2015.

Hyeun Jun Moon, Min Seok Choi, Sa Kyum Kim, and Seung Ho Ryu. Case studies for the evaluation of interoperability between a bim based architectural model and building performance analysis programs. In *Proceedings of 12th Conference of International Building Performance Simulation Association*, volume 2011, 2011.

DD Nebert. Developing spatial data infrastructures: the sdi cookbook, version 2.0.[sl]: Gsdi-technical working group. 2004, 2012.

Romain Nouvel, Jean-Marie Bahu, Robert Kaden, Jerome Kaempf, Piergiorgio Cipriano, Moritz Lauster, Karl-Hainz Haefele, Esteban Munoz, Olivier Tournaire, and Egbert Casper10. Development of the citygml application domain extension energy for urban energy simulation. *Proceedings of Building Simulation 2015*, 2015.

Romain NOUVEL, Robert KADEN, Jean-Marie BAHU, Jerome KAEMPF, Piergiorgio CIPRIANO, Moritz LAUSTER, Joachim BENNER, Esteban MUNOZ, Olivier TOURNAIRE, and Egbert CASPER. Genesis of the citygml energy ade. In *Proceedings of International Conference CISBAT 2015 Future Buildings and Districts Sustainability from Nano to Urban Scale*, number EPFL-CONF-213436, pages 931–936. LESO-PB, EPFL, 2015.

Thomas Nussbaumer and Stefan Thalmann. Status report on district heating systems in iea countries. 2014.

James Pegues. The benefits of 8760 hour-by-hour building energy analysis. *Carrier Software Systems. New York, USA: HVAC Systems Engineer*, 2002.

C Portele. Geography markup language (gml) encoding standard. ogc 07-036, 2007.

Clemens Portele. Ogc geography markup language (gml)–extended schemas and encoding rules, 2012.

M. Ranzinger and G. Gleixner. Digitale 3d-stadtmodelle fr planung und präsentation. In Josef Strobl, Thomas Blaschke, Gerlad Griesebner, and Bernhard Zagel, editors, *Beiträge zum Symposion CORP96*. Wichmann Verlag, 1996.

R Roschlaub and J Batscheider. An inspire-konform 3d building model of bavaria using cadastre information, lidar and image matching. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 747–754, 2016.

Bjorn Sandvik. Using kml for thematic mapping. *Institute of Geography School of GeoSciences. Edinburgh, University of Edinburgh. MSc in Geographical Information Science*, 22, 2008.

Ralph Schildwächter. 3d-stadtmodelle - spielzeug oder arbeitshilfe. *Berlin*, 2005.

Alexandra Stadler and Thomas H Kolbe. Spatio-semantic coherence in the integration of 3d city models. In *Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede*, 2007.

Alexandra Stadler, Claus Nagel, Gerhard König, and Thomas H Kolbe. Making interoperability persistent: A 3d geo database based on citygml. In *3D Geo-Information Sciences*, pages 175–192. Springer, 2009.

Sven Tschirner, Ansgar Scherp, and Steffen Staab. Semantic access to inspire. In *Terra Cognita Workshop*, 2011.

A. Tzoulis. Performance assessment of building energy modelling programs and control optimization of thermally activated building systems. Master's thesis, Delft University of Technology, 2014.

Europäische Union. Richtlinie 2010/31/eu des europäischen parlaments und des rates vom 19. mai 2010 über die gesamtenergieeffizienz von gebäuden (epbd). *Amtsblatt der Europäischen Union*, 53:13–35, 2010.

Detlev Wagner, Mark Wewetzer, Jürgen Bogdahn, Nazmul Alam, Margitta Pries, and Volker Coors. Geometric-semantical consistency validation of citygml models. In *Progress and new trends in 3D geoinformation sciences*, pages 171–192. Springer, 2013.

Parag Wate and Volker Coors. 3d data models for urban energy simulation. *Energy Procedia*, 78:3372 – 3377, 2015. ISSN 1876-6102. doi: http://dx.doi.org/10.1016/j.egypro.2015.11.753. URL `http://www.sciencedirect.com/science/article/pii/S1876610215024856`. 6th International Building Physics Conference, IBPC 2015.

Bruno Willenborg. Simulation of explosions in urban space and result analysis based on citygml city models and a cloud based 3d web client. Masterarbeit, Technische Universitt Mnchen, Mnchen, 2015.

Tim Wilson. Ogc kml, version 2.2. 0. *Open Geospatial Consortium*, 2008.

Zheng Yang and Burcin Becerik-Gerber. Coupling occupancy information with hvac energy simulation: a systematic review of simulation programs. In *Proceedings of the 2014 Winter Simulation Conference*, pages 3212–3223. IEEE Press, 2014.

Wolfgang Zahn. Sonneneinstrahlungsanalyse auf und informationsanreicherung von groen 3d-stadtmodellen im citygml-schema. Masterarbeit, Technische Universitt Mnchen, Mnchen, Feb 2015.

Junqiao Zhao, Jantien Stoter, and Hugo Ledoux. A framework for the automatic geometric repair of citygml models. In *Cartography from pole to pole*, pages 187–202. Springer, 2014.

Xin Zhou. Comparison of building energy modeling programs: Hvac systems. 2014.

A. Appendix A - Mappings

BuildingSurface:Detailed

Attribute	**CityGml**
Name	ThermalBoundary.id OR SurfaceGeometry.id
Surface Type	"Floor" OR "Wall" OR "Ceiling" OR "Roof"
Construction Name	Construction.id
Zone Name	ThermalZone.id
Outside Boundary Condition	"Surface" OR "Outdoors" OR "Ground" OR <blank>
Outside Boundary Condition Object	ThermalBoundary.id OR <blank>
Sun Exposure	"SunExposed" OR "NoSun"
Wind Exposure	"WindExposed" OR "NoWind"
View Factor to Ground	"autocalculate"
Number of Vertices	Count of <gml:pos> elements OR <gml:posList> vertices
Vertex n X-Coordinate	X-Coordinate from <gml:pos> or <gml:posList>
Vertex n Y-Coordinate	Y-Coordinate from <gml:pos> or <gml:posList>
Vertex n Z-Coordinate	Z-Coordinate from <gml:pos> or <gml:posList>

Construction

Attribute	CityGml
Name	AbstractMaterial.id
Outside Layer	Construction.Layer[0].LayerComponent[0].AbstractMaterial.id
Layer2	Construction.Layer[1].LayerComponent[0].AbstractMaterial.id
Layer3	Construction.Layer[2].LayerComponent[0].AbstractMaterial.id
Layer4	Construction.Layer[3].LayerComponent[0].AbstractMaterial.id
Layer5	Construction.Layer[4].LayerComponent[0].AbstractMaterial.id
Layer6	Construction.Layer[5].LayerComponent[0].AbstractMaterial.id
Layer7	Construction.Layer[6].LayerComponent[0].AbstractMaterial.id
Layer8	Construction.Layer[7.LayerComponent[0].AbstractMaterial.id
Layer9	Construction.Layer[8].LayerComponent[0].AbstractMaterial.id
Layer10	Construction.Layer[9].LayerComponent[0].AbstractMaterial.id

DesignSpecification:OutdoorAir

Attribute	CityGml
Name	ThermalZone.id + "_DSOA"
Outdoor Air Method	"Flow/Person"
Outdoor Air Flow per Person	"0.00944"

FenestrationSurface:Detailed

Attribute	CityGml
Name	ThermalBoundary.id OR SurfaceGeometry.id
Surface Type	"Window" OR "Door"
Construction Name	Construction.id
Zone Name	ThermalZone.id
Outside Boundary Condition Object	<blank>
View Factor to Ground	"autocalculate"
Shading Control Name	<blank>
Frame and Divider Name	<blank>
Multiplier	"1"
Number of Vertices	Count of <gml:pos> elements OR <gml:posList> vertices
Vertex n X-Coordinate	X-Coordinate from <gml:pos> OR <gml:posList>
Vertex n Y-Coordinate	Y-Coordinate from <gml:pos> OR <gml:posList>
Vertex n Z-Coordinate	Z-Coordinate from <gml:pos> OR <gml:posList>

ElectricEquipment

Attribute	CityGml
Name	ElectricalAppliances.id
Zone or ZoneList Name	ThermalZone.id
Schedule Name	Schedule.id
Design Level Calculation Method	"EquipmentLevel"
Design Level	ElectricalAppliances.electricalPower
Watts per Zone Floor Area	<blank>
Watts per Person	<blank>
Fraction Latent	ElectricalAppliances.heatDissipation.latentFraction
Fraction Radiant	ElectricalAppliances.heatDissipation.radiantFraction
Fraction Lost	"0"
End-Use Subcategory	"General"

HotWaterEquipment

Attribute	CityGml
Name	DHWFacilities.id
Zone or ZoneList Name	ThermalZone.id
Schedule Name	Schedule.id
Design Level Calculation Method	"Watts/Area"
Design Level	<blank>
Power per Zone Floor Area	DHWFacilities.heatDissipation.totalValue
Power per Person	<blank>
Fraction Latent	DHWFacilities.heatDissipation.latentFraction
Fraction Radiant	DHWFacilities.heatDissipation.radiantFraction
Fraction Lost	"0"
End-Use Subcategory	"General"

Lights

Attribute	CityGml
Name	LightingFacility.id
Zone or ZoneList Name	ThermalZone.id
Schedule Name	Schedule.id
Design Level Calculation Method	"LightingLevel"
Lighting Level	LightingFacility.electricalPower
Watts per Zone Floor Area	<blank>
Watts per Person	<blank>
Return Air Fraction	"0"
Fraction Radiant	LightingFacility.heatDissipation.radiantFraction
Fraction Visible	"0.18"
Fraction Replaceable	"0"
End-Use Subcategory	"General"
Return Air Fraction Calculated from Plenum Temperature	"No"

Material

Attribute	CityGml
Name	AbstractMaterial.id
Roughness	"MediumRough"
Thickness	LayerComponent.Thickness
Conductivity	SolidMaterial.conductivity OR 0 (GAS)
Density	SolidMaterial.density OR 0 (Gas)
Specific Heat	SolidMaterial.specificHeat OR 100 (Gas)

Material:NoMass

Attribute	CityGml
Name	AbstractMaterial.id
Roughness	"MediumSmooth"
Thermal Resistance	R value calculated from Construction.uValue (1/uValue)

OutdoorAir:Node

Attribute	CityGml
Name	ThermalZone.id + "_OutdoorAir"
Height Above Ground	"-1.0"

People

Attribute	CityGml
Name	Occupants.id
Zone or ZoneList Name	ThermalZone.id
Number of People Schedule Name	Schedule.id
Number of People Calculation Method	"People" OR "Area/Person"
Number of People	Occupants.numberOfOccupants
People per Zone Floor Area	<blank>
Zone Floor Area per Person	<blank>
Fraction Radiant	Occupants.heatDissipation.radiantFraction
Sensible Heat Fraction	"autocalculate"
Activity Level Schedule Name	Schedule.id
Carbon Dioxide Generation Rate	"0.0000000382"
Enable ASHRAE 55 Comfort Warnings	"No"
Mean Radiant Temperature Calculation Type	"ZoneAveraged"

Schedule:Compact

Attribute	CityGml
Name	Schedule.id
Schedule Type Limits Name	"Fraction" OR "Any Number" OR "Temperature" OR "On/Off" OR "Control Type" OR "Humidity"
Field n	Value according to attribute Schedule Type Limits Name

WindowMaterial:SimpleGlazingSystem

Attribute	CityGml
Name	AbstractMaterial.id
U-Factor	Construction.uValue
Solar Heat Gain Coefficient	Construction.OpticalProperties.Transmittance.Fraction

Zone

Attribute	CityGml
Name	ThermalZone.id
Direction of Relative North	"0.0"
X Origin	"0.0"
y Origin	"0.0"
Type	"0.0"
Multiplier	"1"

ZoneInfiltration:DesignFlowRate

Attribute	CityGml
Name	ThermalZone.id + "_Infiltration"
Zone or ZoneList Name	ThermalZone.id
Schedule Name	Schedule.id
Design Level Calculation Method	"AirChanges/Hour"
Design Flow Rate	<blank>
Flow per Zone Floor Area	<blank>
Flow per Exterior Surface Area	<blank>
Air Changes per Hour	"0.3"
Constant Term Coefficient	"1"
Temperature Term Coefficient	<blank>
Velocity Term Coefficient	<blank>
Velocity Squared Term Coefficient	<blank>

ZoneControl:Humidistat

Attribute	CityGml
Name	ThermalZone.id + _Humidistat
Zone Name	ThermalZone.id
Humidifying Relative Humidity Setpoint Schedule Name	Schedule.id
Dehumidifying Relative Humidity Setpoint Schedule Name	Schedule.id

ZoneControl:Thermostat

Attribute	CityGml
Name	ThermalZone.id + "_Thermostat"
Zone or ZoneList Name	ThermalZone.id
Control Type Schedule Name	Schedule.id
Control 1 Object Type	"ThermostatSetpoint:DualSetpoint"
Control 1 Name	"Thermostat_AllZones"

ZoneHVAC:EquipmentConnections

Attribute	CityGml
Zone Name	ThermalZone.id
Zone Conditioning Equipment List Name	ThermalZone.id + "_Equipment"
Zone Air Inlet Node or NodeList Name	ThermalZone.id + "_Inlet"
Zone Air Exhaust Node or NodeList Name	<blank>
Zone Air Node Name	ThermalZone.id + "_AirNode"
Zone Return Air Node Name	ThermalZone.id + "_Outlet"

ZoneHVAC:EquipmentList

Attribute	CityGml
Name	ThermalZone.id + "_Equipment"
Zone Equipment 1 Object Type	"ZoneHVAC:IdealLoadsAirSystem"
Zone Equipment 1 Name	ThermalZone.id + "_ZoneHVAC"
Zone Equipment 1 Cooling Sequence	"1"
Zone Equipment 1 Heating or No-Load Sequence	"1"

ZoneHVAC:IdealLoadsAirSystem

Attribute	CityGml
Name	ThermalZone.id + "_ZoneHVAC"
Availability Schedule Name	Schedule.id
Zone Supply Air Node Name	ThermalZone.id + "_Inlet"
Zone Exhaust Air Node Name	<blank>
Maximum Heating Supply Air Temperature	"50.0"
Minimum Cooling Supply Air Temperature	"20.0"
Maximum Heating Supply Air Humidity Ratio	"0.012"
Minimum Cooling Supply Air Humidity Ratio	"0.0077"
Heating Limit	"NoLimit"
Maximum Heating Air Flow Rate	"autosize"
Maximum Sensible Heating Capacity	<blank>
Cooling Limit	"NoLimit"
Maximum Cooling Air Flow Rate	"autosize"
Maximum Total Cooling Capacity	<blank>
Heating Availability Schedule Name	<blank>
Cooling Availability Schedule Name	<blank>
Dehumidification Control Type	"None"
Cooling Sensible Heat Ratio	"0.7"
Humidification Control Type	"None"
Design Specification Outdoor Air Object Name	"OutdoorAirOffice"
Outdoor Air Inlet Node Name	ThermalZone.id + "_OutdoorAir"
Demand Controlled Ventilation Type	"None"
Outdoor Air Economizer Type	"NoEconomizer"
Heat Recovery Type	"None"
Sensible Heat Recovery Effectiveness	"0.90"
Latent Heat Recovery Effectiveness	"0.65"

B. Appendix B - Activity Diagrams

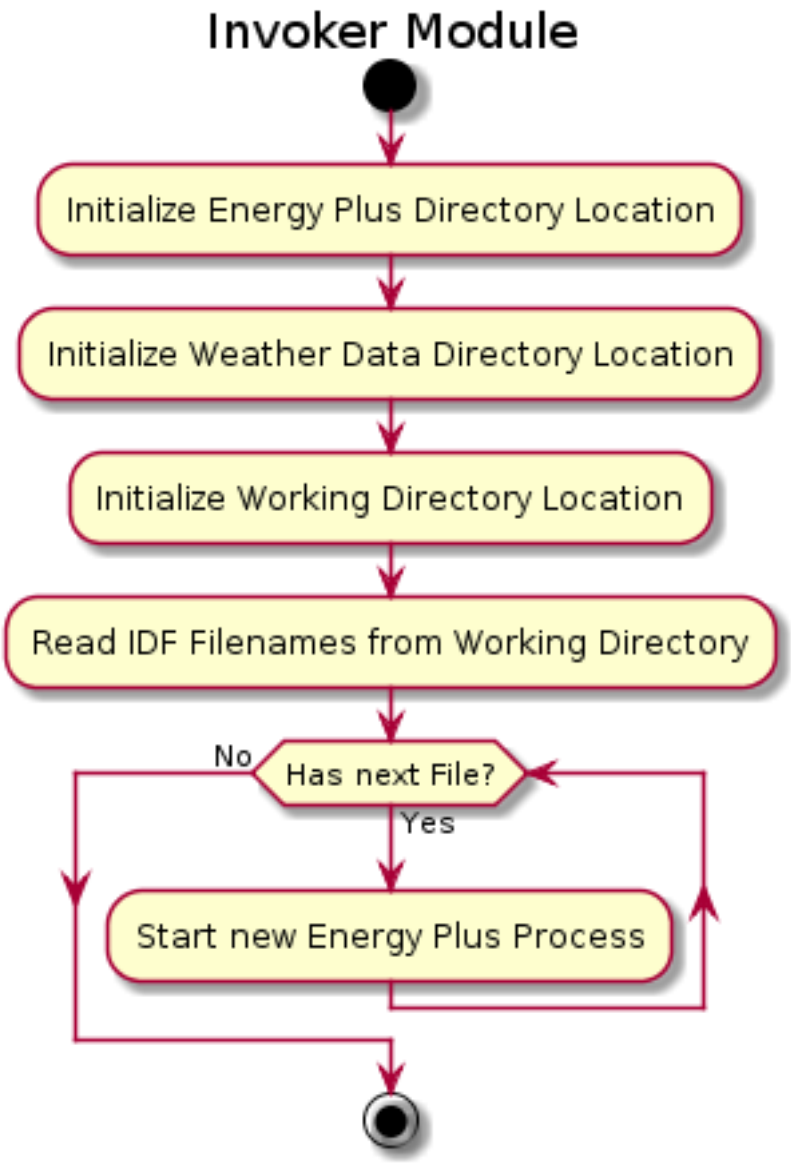

Figure B.1.: Activity Diagram Invoker Module

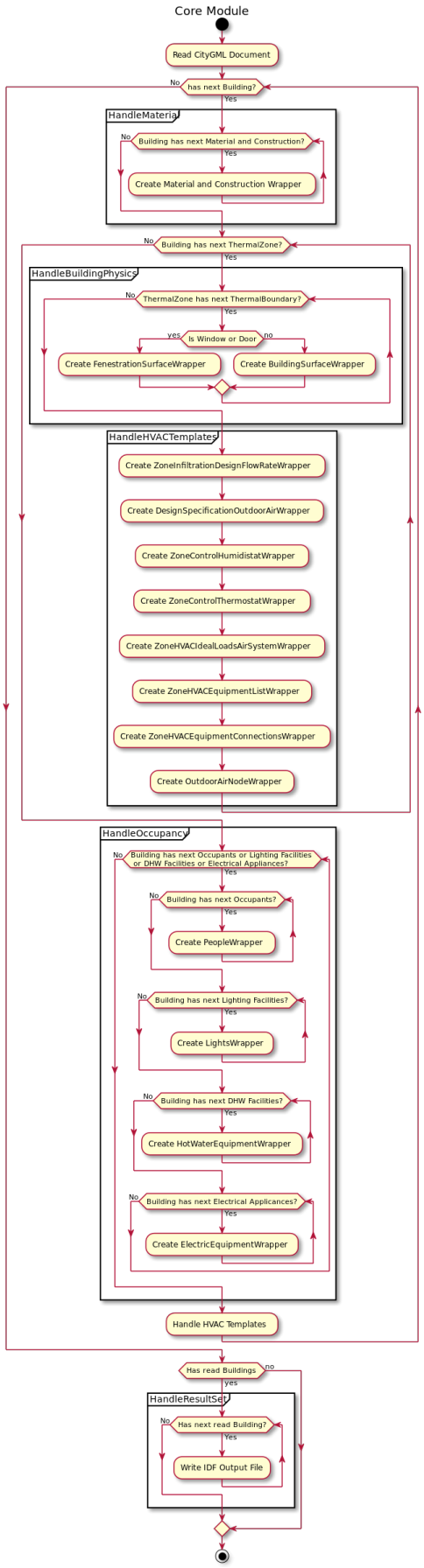

Figure B.2.: Activity Diagram Core Module

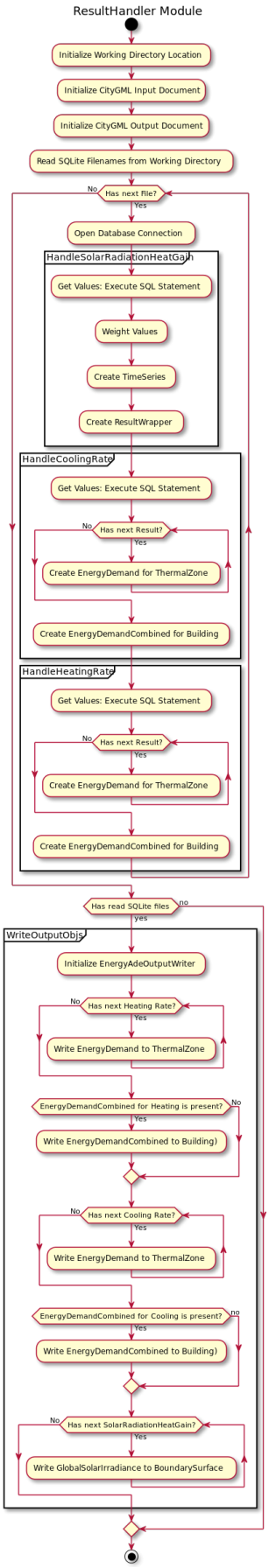

Figure B.3.: Activity Diagram Result Handler Module

C. Appendix C - SQL Statements

```
SELECT rdd.KeyValue, rd.Value, rdd.ReportingFrequency, rdd.Units
  FROM ReportData rd
  LEFT OUTER JOIN ReportDataDictionary rdd
  ON rd.ReportDataDictionaryIndex = rdd.ReportDataDictionaryIndex
  WHERE rd.ReportDataDictionaryIndex IN
  (
    SELECT ReportDataDictionaryIndex
    FROM ReportDataDictionary
    WHERE Name = 'Surface Outside Face Sunlit Area'
    ORDER BY ReportDataDictionaryIndex ASC
  )
  ORDER BY rd.ReportDataDictionaryIndex, rd.TimeIndex ASC
```

Listing C.1: SQL Query for EnergyPlus output variable "Surface Outside Face Incident Solar Radiation Rate per Area"

```
SELECT rdd.KeyValue, rd.Value, rdd.ReportingFrequency, rdd.Units
  FROM ReportData rd
  LEFT OUTER JOIN ReportDataDictionary rdd
  ON rd.ReportDataDictionaryIndex = rdd.ReportDataDictionaryIndex
  WHERE rd.ReportDataDictionaryIndex IN
  (
    SELECT ReportDataDictionaryIndex
    FROM ReportDataDictionary
    WHERE Name = 'Zone Ideal Loads Zone Total Cooling Rate'
    ORDER BY ReportDataDictionaryIndex ASC
  )
  ORDER BY rd.ReportDataDictionaryIndex, rd.TimeIndex ASC
```

Listing C.2: SQL Query for EnergyPlus output variable "Zone Ideal Loads Zone Total Cooling Rate"

```
SELECT SUM(rd.Value) Value, rdd.ReportingFrequency, rdd.Units
  FROM ReportData rd
  LEFT OUTER JOIN ReportDataDictionary rdd
  ON rd.ReportDataDictionaryIndex = rdd.ReportDataDictionaryIndex
  WHERE rd.ReportDataDictionaryIndex IN
  (
    SELECT ReportDataDictionaryIndex
    FROM ReportDataDictionary
    WHERE Name = 'Zone Ideal Loads Zone Total Cooling Rate'
    ORDER BY ReportDataDictionaryIndex ASC
   )
  GROUP BY rd.TimeIndex
  ORDER BY rd.ReportDataDictionaryIndex, rd.TimeIndex ASC
```

Listing C.3: SQL Query for combinend EnergyPlus output variable "Zone Ideal Loads Zone Total Cooling Rate"

```
SELECT rdd.KeyValue, rd.Value, rdd.ReportingFrequency, rdd.Units
  FROM ReportData rd
  LEFT OUTER JOIN ReportDataDictionary rdd
  ON rd.ReportDataDictionaryIndex = rdd.ReportDataDictionaryIndex
  WHERE rd.ReportDataDictionaryIndex IN
  (
    SELECT ReportDataDictionaryIndex
    FROM ReportDataDictionary
    WHERE Name = 'Zone Ideal Loads Zone Total Heating Rate'
    ORDER BY ReportDataDictionaryIndex ASC
  )
  ORDER BY rd.ReportDataDictionaryIndex, rd.TimeIndex ASC
```

Listing C.4: SQL Query for combinend EnergyPlus output variable "Zone Ideal Loads Zone Total Heating Rate"

```
SELECT SUM(rd.Value) Value, rdd.ReportingFrequency, rdd.Units
  FROM ReportData rd
  LEFT OUTER JOIN ReportDataDictionary rdd
  ON rd.ReportDataDictionaryIndex = rdd.ReportDataDictionaryIndex
  WHERE rd.ReportDataDictionaryIndex IN
  (
    SELECT ReportDataDictionaryIndex
    FROM ReportDataDictionary
    WHERE Name = 'Zone Ideal Loads Zone Total Heating Rate'
    ORDER BY ReportDataDictionaryIndex ASC
  )
  GROUP BY rd.TimeIndex
  ORDER BY rd.ReportDataDictionaryIndex, rd.TimeIndex ASC
```

Listing C.5: SQL Query for combinend EnergyPlus output variable "Zone Ideal Loads Zone Total Heating Rate"

D. Glossary

3D Three Dimensional

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BIM Building Information Modeling

COLLADA COLLAborative Design Activity

DTM Digital Terrain Model

FME Feature Manipulation Engine

GIS Geographic Information System

GML Geography Markup Language

HVAC Heating, Ventilation and Air Conditioning

IDD Input Data Dictionary

IDF Input Data File

IFC Industry Foundation Classes

INSPIRE Infrastructure for Spatial Information in the European Community

ISO International Organization for Standardisation

JAXB Java Architecture for XML Binding

JVM Java Virtual Machine

KML Keyhole Markup Language

Lidar Light Detection and Ranging

LOD Level of Detail

OGC Open Geospatial Consortium

SDI Spatial Data Infrastructure

SQL Simple Query Language

UML Unified Modeling Language

XML Extensible Markup Language

XSD XML Schema Definition

List of Figures

List of Tables

Listings