CITYGML IMPORT AND EXPORT FOR DYNAMIC BUILDING PERFORMANCE SIMULATION IN MODELICA

Peter Remmen, Moritz Lauster, Michael Mans, Tanja Osterhage and Dirk Müller

RWTH Aachen University, E.ON Energy Research Center, Institute for Energy Efficient Buildings and Indoor Climate

ABSTRACT

Energy supply of buildings in the urban context currently undergoes significant changes. To consider these changes in the energy systems and the buildings themselves, dynamic simulation, in particular the heat demand of multiple buildings, is one key element. The presented work introduces a methodology to use CityGML data sets for dynamic building simulation in Modelica. The framework includes the import of CityGML and the extraction of available geometric and semantic information. This information is mapped to a software tool called *TEASER*, which is available open-source (https://github.com/RWTH-EBC/TEASER). The tool extends available information with statistical data and generates a Modelica model. To proof the concept, we apply the methodology on two use cases. The first use case shows the general workflow with a fictitious CityGML data set. The second use case demonstrates the capability to handle large data sets, using a CityGML file with more than 2,800 buildings with different level of detail.

INTRODUCTION

Higher energy standards and efficiency requirements for buildings impose significant changes in the energy supply of buildings in the urban context. One approach is the study of entire city districts; what allows investigating energy production, distribution and consumption as an integral system. For an analysis of interactions within this system, static calculation methods are insufficient, as they often provide monthly or yearly values. It is for this reason that all three areas of the urban energy system increasingly use dynamic simulation models. In this way, the energy flows are calculated with a high temporal resolution and dynamic effects like heat storage are taken into account. Especially integration of renewable energy sources requires a high temporal resolution of consumption and production. The occurring time shifts have to be identified to work out suitable solutions. The dynamic heat demand of buildings is a key element in the consideration of city districts. Previous work deals with the development and validation of a dynamic building model in the non-proprietary modelling language Modelica (Lauster et al. 2014c; Lauster et al. 2014b; Lauster et al. 2014a; Lauster et al. 2015). The modelling approach of the building model follows a design driven reduced order model. One main advantage of Modelica is the object-oriented design that allows combining and fast prototyping of energy production, distribution and consumption in one model (Wetter, Bonvini, and Nouidui 2015). Other examples for dynamic heat demand calculation on district scale can be found in (Reinhart and Cerezo Davila 2016; Robinson et al. 2009; Kämpf and Robinson 2007). To parameterize design driven models, necessary building characteristics must be handed over to the building model. However, detailed information on urban scale is sparse. One approach is to define archetype buildings that are used to extend basic building parameters to a fully described archetype building (Mata et al. 2014). Basic geometrical and semantic building parameters can be extracted from the urban information model CityGML. Similar work can be found in (Nouvel et al. 2015a; Robinson et al. 2009). This paper provides an overview of a methodology for automated Modelica model generation from CityGML as data source. The main benefit of this methodology is a high modularity that allows extending the tool in every step and the applicability to the modelling language Modelica. The first section describes the used data format CityGML, its energy related extension EnergyADE and the Modelica Reduced Order Model. The methodology section illustrates the process, controlled by the Python software tool *TEASER*. Import, analysis, statistical enrichment, model generation and export are presented systematically. Two use cases will present the application of the framework. Finally, we discuss the limitations and possible future research work of the shown process.

CITYGML AND SIMULATION MODEL

CityGML and EnergyADE

CityGML is an information model within the urban context. It is an international standard developed by the Open Geospatial Consortium (OGC) (Gröger et al. 2012) and serves for representation and exchange of 3D city models. CityGML uses the Geography Markup Language (GML) for representation of data and bases on the Extensible Markup Language (XML). XML is a wide spread object-oriented language especially designed for robust data storage and exchange (Miller, Hersberger, and Jones 2013) and uses schema definitions (XML Schema Definition: XSD) to define the structure of an XML data set. Schema definitions specify the hierarchy and relationship between objects in the XML data set and define data types (e.g. string/boolean/integer etc.) and even units of attributes. The possibility to define the structure of individual data sets XSD enables the language XML for interoperability between different applications. The current schema of CityGML is version 2.0 and is available online (CityGML 2016). CityGML is not limited to geometrical data; it covers as well some semantic information of objects (properties and features) (Gröger and Plümer 2012). For this purpose, the information model of CityGML divides into several thematic modules (e.g. vegetation or transportation). Different namespaces represent these modules in the XML and XSD schema. However, in this paper we focus on the building module. CityGML provides the possibility to store data in discrete Level of Detail (LoD 0 to LoD 4). While LoD 0 represents only the footprint area of a building, LoD 4 is a detailed 3D model with interior structures. However, such information is rare on urban scale. For Germany, we observed that LoD 1 and LoD 2 are available for few cities (Bezirksregierung Köln 2016; Stadt Potsdam). While LoD 1 describes buildings as boxes (in LoD 1 roofs are always flat and horizontal), LoD 2 adds more information for exterior walls to the building. CityGML in LoD 2 distinguishes between wall, roof and ground surfaces as boundary surfaces of the building. Besides geometrical data, the building module provides the possibility to add additional attributes to the building. Some examples are function (usage), year of construction, number and height of storeys. Although CityGML can store all basic information for city information models, special applications may require additional information. For this purpose, Application Domain Extensions (ADE) extend the CityGML schema. An ADE can be interpreted as an additional module of CityGML with an own XML namespace. Popular examples for ADEs are Noise and UtilityNetwork ADE (Becker, Nagel, and Kolbe 2016). The ADE concept has two mechanisms to extend CityGML:

- New attributes and features are added to existing CityGML classes
- Definition of new, ADE-specific classes, these classes may extend existing CityGML classes

Currently an international consortium develops an ADE for the application of (dynamic) building performance simulation (Nouvel et al. 2015b). The current version of the Energy ADE is 0.6.0; it is available as open source (CityGML EnergyADE 2016). The Energy ADE is divided into four modules: *BuildingPhysics* module, *Material* module, *EnergySystem* module, *Occupancy* module. The four modules offer the possibility to store data at different level of detail required to calculate the energy flows in buildings on urban scale. Thus the Energy ADE keeps the flexibility of CityGML and is applicable to different calculation methods e.g. from static analysis of monthly consumption to dynamic BPS. The core of the ADE is the *BuildingPhysics* module. It covers the definition of the building with thermal boundaries and thermal zones and directly links to the *Building* module of CityGML. The *Material* module of the Energy ADE contains the description of all building construction elements. It is possible to add basic values like the overall U-Value, as well as the definition of the exact construction of a wall, layer by layer. Information of the usage type or exact schedules are described by the *Occupancy* module. Energy demands and basic energy supply technologies are described with help of the *EnergySystem* module.

Reduced Order Model

Dynamic simulation models to determine heat demand on urban scale need to find a trade-off between accuracy and computational effort. We are using a Reduced Order Model (ROM) from Modelica library AixLib. The AixLib is available open-source at https://github.com/RWTH-EBC/AixLib. Modelica is a free, open-source and non-proprietary modelling language for complex technical systems. Equation-based and object oriented blocks compose the models while acausal interfaces for different physical phenomena (e.g. heat transfer) connect the blocks. The International Energy Agency's (IEA) Annex 60 project "New Computational Tools for Building Performance Simulation" is part of the Energy in Buildings and Communities Programme (IEA-EBC), and coordinates the development of BPS in Modelica. The described model was explicitly developed to model and simulate multiple buildings on urban scale (Lauster et al. 2014c). To keep calculation time as low as possible, the ROM focuses on predominant effects, this includes neglecting physical phenomena of lower importance to the building's heat demand. The ROM is based on the German Guideline VDI 6007-1 and is constantly under development, which leads to three different versions of the model (Lauster et al. 2014b; Lauster et al. 2014a). The model clusters all building elements of the same type (e.g. walls facing to the outdoor environment). This results in a lumped parameter model with two Resistance-Capacitance-Combinations for exterior (2R1C), interior (1R1C) walls, and one resistance for windows. RC-Combinations are used to calculate thermal conduction and storage effects. The interior walls are considered as adiabatic and serve as heat storage. The advantage of the reduced order model is a comparable low calculation time, due to highly reduced number of state variables. However, simplification always results in lower accuracy of the model. Nevertheless different applications already proofed the applicability of the model on urban scale (Fuchs et al. 2015; Schiefelbein et al. 2015).

METHODOLOGY

The following chapter presents the conversion process from a CityGML data set to a ready-to-use Modelica model. To accomplish this transformation we identified three necessary steps:

1. Import and analysis of CityGML data set
2. Enrich the CityGML data set
3. Generate Modelica code

A fourth and optional step in our methodology is the export of the enriched data set into a valid CityGML file, for further use in other software tools.

Figure 1 displays the mentioned steps. A self-developed software tool called *TEASER “Tool for Energy Analysis and Simulation for Efficient Retrofit”* controls the process. The tool derives from the *Retrofit Matrix*, developed by (Hillebrand et al. 2014). To meet the requirements of each step, we developed an internal object-oriented data model for *TEASER*, its simplified data model is shown in Figure **2**. Other information models on building scale influence the hierarchical structure of *TEASER* in addition to CityGML, particular examples are *gbXML* and *SimModel*, two information models with a focus on energy analysis of buildings (Roth 2016; O'Donnell et al. 2011). In contrast to these examples, *TEASER* does not provide a geometrical representation of the building, thus requires a simplified data model. The root element of *TEASER* is the *Project* class, at the same time this class serves as the Application Programming Interface (API) for *TEASER*. API functions control different processes in *TEASER* (e.g. import and export). In addition, the *Project* class has a list with all *Building* instances as an attribute. The *Building* class collects information on building level. On the one hand, these are basic information that may be used for data enrichment (like construction year, total net-leased area or address) on the other hand; a container stores all *ThermalZone* instances. The *ThermalZone* class provides zone related parameters like area, volume and infiltration rate. In addition, three lists store exterior walls (including roofs and ground floors), interior walls (including floor and ceiling) and windows. For each zone, it is possible to define different use conditions, which may provide set temperatures, occupancy profiles and other user related attributes. As one main goal of our methodology is to generate lumped parameter models, the *ThermalZone* object provides functions to calculate these parameters. It is easily possible to extend the *ThermalZone* class with further calculation methods, e.g. for further lumped parameter models. All *Wall* and *Window* objects inherit from the generic class *BuildingElement*. Orientation, tilt, area and physical properties like thermal resistances are stored or calculated in the *BuildingElement* object. Each *BuildingElement* has a container for storing arbitrary number of *Layer* classes. *Layer* instances store information regarding the thickness and the material. All hierarchical connected elements are linked by a parent-child relationship. This enables to access data of related objects with a simple attribute call. *TEASER* is written in the programming language Python. Python has an object-oriented, functional and structural character. It has a strong emphasis on code readability, which opens *TEASER* also to non-programmers. Furthermore, there are numerous packages for scientific and engineering applications, which are easy to import into a Python programme. *TEASER* itself is also designed as a Python package to improve the usability.

Import and analysis of CityGML data set

The import of a CityGML data set includes two tasks. First, a function loads the data set into the Python environment and extracts the available information; the second step consists of a basic geometric pre-processor to extract orientation, tilt and area of all surfaces.

As described in the first section, CityGML data sets are saved in an XML file. Different possibilities and

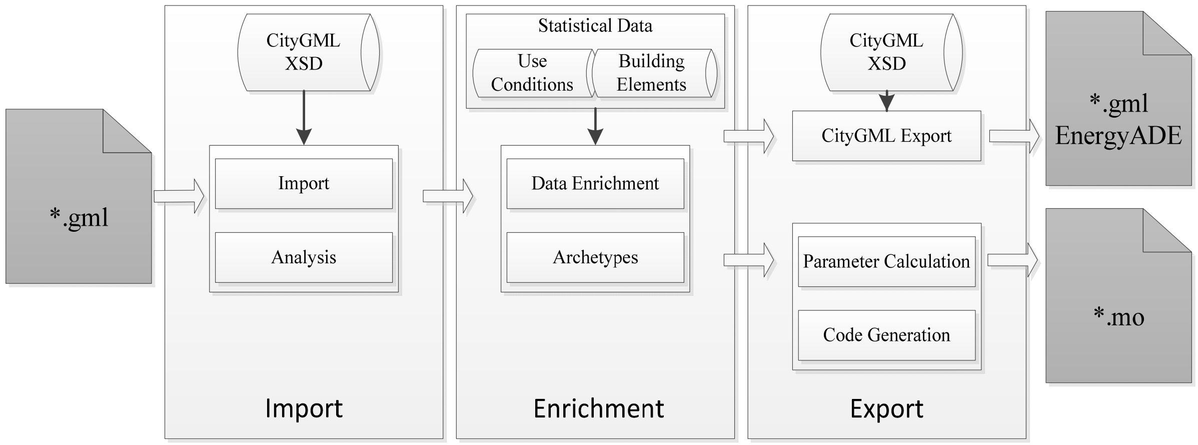

Figure 1: Methodology overview to generate Modelica models from CityGML data sets

packages already exist to load and parse XML data

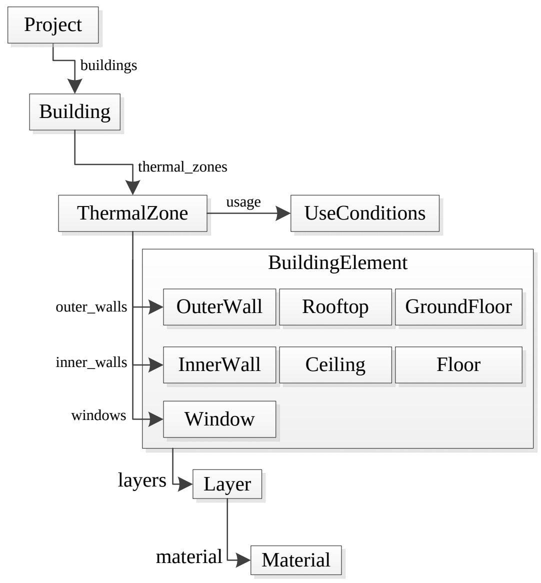

Figure 2: simplified UML diagram of *TEASERs* data model

sets in Python. The most basic way is to use standardised APIs, like Simple API for XML (SAX) or Document Object Model (DOM) API. Other possibilities are special packages developed for parsing XML. Famous examples are *xml.etree.ElementTree*, from the Python standard library or *lxml*, without being exhaustive (Behnel, S: et. al. 2016; Python Software Foundation 2016). However, for the import of CityGML we are using the Python package *PyXB* to generate binding classes in Python corresponding to the CityGML XSD schema definition (Bigot 2016). Binding classes are an exact mapping of every CityGML class to a Python class. Compared to the above-mentioned APIs and packages, the binding provides access to the data set using Python objects and attributes instead of generic elements, attributes and text. This results in a clear structured code that aligns to the CityGML data set. More over the same objects are used for a bi-directional conversion from CityGML to Python and vice versa. *PyXB* creates binding classes automatically. Automatic generated code helps to access large schema definitions like CityGML more efficiently. Furthermore, *PyXB* validates automatically input and output documents against the XSD schema. Validation in this context means to check values to be in the correct range or units. As mentioned in the section before, the building module holds semantic and geometric information of a building. The tool focusses (but is not limited) on extracting following attributes.

Semantic data:

- *bldg:function*
- *bldg:yearOfConstruction*
- *bldg:measuredHeight*
- *bldg:storeysAboveGround*
- *bldg:storeyHeightsAboveGround*
- *bldg:roofType*

Geometric data:

- *bldg:lod1Solid*
- *bldg:lod2Solid*
- *bldg:boundedBy (only for LoD 2)*

The methodology uses semantic data by mapping them to *TEASERs* internal data model, without any further processing. Depending of data availability, the tool uses semantic data for data enrichment. The next section explains this in more detail.

There is more than one option to model buildings in the different LoD’s. However, Gröger and Plümer recommend modelling LoD 1 with *bldg:lod1Solid* and LoD 2 with *bldg:lod2Solid* and *bldg:boundedBy*. Four or more 3D coordinates, with the normal pointing always outside of the building, should define a polygon. With this convention, the import calculates orientation, tilt and area of the surface and maps it to *BuildingElements* of the internal data model of *TEASER*. For LoD 1, we developed following simple rules: vertical surfaces (tilt = 90°) correspond to exterior walls, horizontal (tilt = 0°) depending on their normal to roof or ground surfaces. If the building is represented in LoD 2, the tool maps the surfaces according to the definition in CityGML.

Enrichment of the data set

CityGML provides geometrical and a few semantic data of a building. This data is not sufficient for dynamic BPS. Sparse data is a typical problem for dynamic simulation on urban scale. Therefore previous work showed the possibility to use archetype buildings defined by a limited number of elementary building parameters to estimate enclosure area and usage zones inside the building (Fuchs et al. 2015; Hillebrand et al. 2014; Schiefelbein et al. 2015). With the statistical data enrichment, *TEASER* adds physical properties, like wall construction layer by layer to the internal data model (Loga et al. 2005). These elementary parameters are building type (e.g. office), year of construction, floor height, number of floors and net leased area. Since the CityGML data set provides exact enclosure area, we only use parts of the archetype building generation. Precisely this means that we are using the exact enclosing surface of the building, but statistical data for physical properties based on the year of construction and usage zones inside the building based on the function of the building. Additionally we need to estimate the share of windows in the enclosing area. In a perfect scenario, the CityGML data set holds the number or the height of floors, these attributes are crucial to estimate the amount of interior walls of the building and thus its thermal mass. If they are not available, we estimate the height of floors, depending on the building type. Based upon the function of the building, *TEASER* adds basic profiles for different usage zones for internal loads (machines and lighting) and user occupancy. These profiles are taken from DIN V 18599 and SIA 2024 (Deutsches

Institut für Normung; Swiss Society of Engineers and Architects). Currently *TEASER* supports three archetype buildings for distinction of different usage zones: residential, office and laboratory. However, the object-oriented character of *TEASER* provides the possibility to define more archetype buildings.
Another feature of *TEASER* is to retrofit buildings. Depending on the year of retrofit, windows are replaced and an additional insulation layer is added to outer walls. Thermal properties of the window and the thickness of the insulation layer are set according to the retrofit standard in Germany of the year of retrofit. For the year 2015 this is the EnEV 2014 that describes a U-Value of 0.24 W/(m²*K) for vertical outer walls and 1.3 W/(m²*K) for windows.

Generate Modelica Code

The last step in our methodology is the generation of ready-to-use Modelica code. This task divides into the calculation of the lumped parameters for the used reduced order model and the textual output of the code itself. The determination of parameters for the simulation model needs calculation on wall layer, wall, zone and building level. The object-oriented character of *TEASER* provides a useful basis to calculate these values and remains flexible enough to integrate calculation methods for other building models, e.g. other lumped parameter model that are currently under development in the Annex 60 project. The reduced order model of AixLib is a 0D model and modelled with object-oriented character. The code generation benefits of these characteristics, as it only needs to instantiate the objects and set all necessary parameters. We investigated two different concepts to generate text as output of a computer program. The first approach is to use print-statements. Print-statements are useful for short outputs that do not need any further conversion, e.g. for syntax. The second possibility is to use templates. In templates, placeholders define the location of the parameter within a pre-built structure. We are using Mako template engine (Mayer 2016). The template engine replaces the placeholder with the actual value. In templates, it is possible to define control structures, this helps for example to convert lists in Python syntax into the correct syntax for Modelica. Manipulation of the actual data is not necessary in this case. Another advantage is a clearly defined interface, so if the model changes we do not need to touch the Python code, but only adapt the template.

Export CityGML with enriched data

One advantage of standardised exchange specifications is interoperability between different tools. We provide two possibilities to export *TEASER* data into a *CityGML* file. The first option is to create a new *CityGML* file; this is useful for buildings created in TEASER manually or from other input sources (e.g. databases). The second option is to extend an existing CityGML file with additional information. *TEASER* adds information about the construction of each building element and about the thermal zones to the data set. TEASER uses following EnergyADE classes for describing building elements (e.g. *energy:Construction*, *energy:Layer*) and thermal zones (e.g. *energy:ThermalZone*, *energy:ThermalBoundarySurface*). Depending on the Level of Detail, this information is stored in different objects in CityGML. Both LoD 1 and LoD 2 buildings are extended by a number of *energy:ThermalZone*. The EnergyADE thermal zone saves attributes like zone area and infiltration rate. Every thermal zone is bounded by *energy:thermalBoundarySurface* objects. For buildings represented in LoD 2, the export function adds construction details directly to *bldg:boundedBy*. In the thermal boundary surface of the thermal zone, these constructions are linked by xlink:href functions. This is a special mechanism of XML to reference same instances to different objects in a data set. This has the advantage that the same construction needs to be described only once. For LoD 1 the construction details are added to the thermal boundary surface of the EnergyADE into the thermal zone.

WORKFLOW DEMONSTRATION

Two use cases prove the concept of importing building data from CityGML data sets, enrich the data and generate Modelica code.
The first fictitious use case 1 consists of six buildings, with LoD 1 and LoD 2 representations. For statistical enrichment, we assume the availability of function, number of storeys and year of construction of the building. The authors are aware that this may not be often the case. The conclusion discusses the limitations in more detail. Figure 3 shows a screenshot of the building setup. First, the CityGML is imported into *TEASER*; building parameters are set according to the function and year of construction with the help of the data enrichment

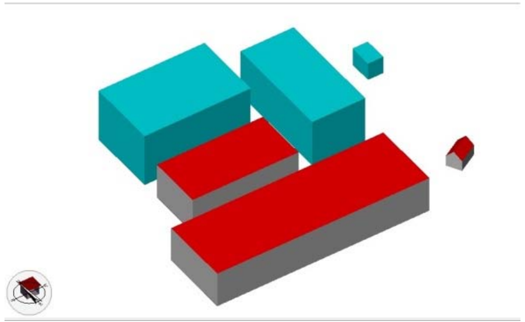
Figure 3: Screenshot of use case 1 shown in FZK viewer (http://www.iai.fzk.de/www-extern/index.php?id=1931)

module of *TEASER*. Finally, the tool generates Modelica code. With the help of the BuildingsPy package, the simulation of all buildings starts automatically. After simulation of the reference case, we retrofit all buildings with a year of retrofit in 2015, generate the updated models, and simulate the scenario again. The export module saves the extended data set into a valid CityGML file, with use of the EnergyADE.

The second use case is a CityGML example file from Bad Godesberg, Bonn in Germany (Bezirksregierung Köln 2016). This file contains 2,897 buildings or building parts (e.g. terraced houses). 31 buildings in LoD 1, 1,793 in LoD 2 and 1,073 in LoD 2 modelled as building parts. This data set provides no information about construction year, function or number of floors. Therefore, year of construction and function are set to random parameters. We are estimating the height of each floor. With this estimation and the height of the building, the number of floors is calculated.

RESULTS

The main result of this paper is a flexible process to generate dynamic simulation models in Modelica. Both use cases are not intended to show reliable simulation results. use case 1 demonstrates each step of the process in a comprehensible way. Use case 2 shows the capability of the tool to load and handle a large number of buildings.

Use case 1

Listing 1 shows the necessary Python code to execute all steps through *TEASER* API. Note that the automation of simulation with BuildingsPy is neglected in this example. Simulation results of one residential building in original and retrofitted condition are shown in Figure 4. As mentioned before, these results are mainly to demonstrate the use of *TEASER*. Another outcome is the extended CityGML data set. Listing 2 shows a simplified representation of fully described building element in LoD 2.

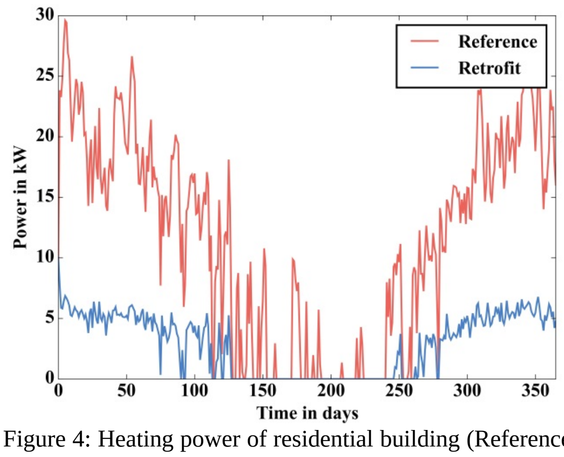

Figure 4: Heating power of residential building (Reference: 1950, Retrofit: 2015)

Use case 2

Due to the unknown year of construction and function of the building, the uncertainty in simulation results for Use case 2 would be massive. However, the main finding of this use case is that the tool is capable of handling a large number of buildings. The amount of time for the different steps is listed in Table 1. Loading the CityGML file into the binding classes accounts by far the largest proportion share on time consumption. However, taking the number of buildings into account, this is still acceptable.

Listing 1: Process control through TEASER API

```
from teaser.project import Project
prj = Project(load_data=True)
prj.load_gml_energy("BSO_2016.gml")
prj.calc_all_buildings("ebc")
prj.generate_model([...])
prj.retrofit_all_buildings(2015)
prj.generate_model([...])
prj.extend_gml("BSO_2016.gml")
```

Listing 2: CityGML example of a building element with use of EnergyADE

```
<bldg:Building>
  <bldg:boundedBy>
    <bldg:GroundSurface gml:id="b_ground">
      <energy:construction>
          <energy:Construction>
            <energy:layer>
              <energy:Layer>
                 [...]
              </energy:Layer>
            </energy:layer>
          </energy:Construction>
      </energy:construction>
    </bldg:GroundSurface>
  </bldg:boundedBy>
</bldg:Building>
```

Table 1: Time consumption of different process steps

Process	**Time**
Import data	231 sec
Analyse data	35 sec
Calculate parameters	16 sec
Generate code	71 sec

LIMITATIONS

The major limitations in the presented methodology are:

CityGML data set

- CityGML data sets usually contain topological errors
- CityGML data sets are usually not entirely complete, especially regarding semantic information (e.g. year of construction)
- Several different options to model buildings and building elements (although these options might not be recommended)
- Only supporting LoD 1 and LoD 2 in import function for geometric calculation

While the quality of CityGML data by means of correct topology and information density is part of other research fields, we concentrate on enhancing the import to be more robust. This includes the implementation of a geometric processor that can handle different types of modelling approaches in CityGML. Further we need to clarify if the implementation of LoD 3 and LoD 4 is feasible in this context.

Data enrichment

- Only limited number of type buildings with limited number of subcategories (Residential, Office/Laboratory)
- Statistical data only for German building stock, no uncertainty in material properties
- User behaviour as static daily profiles

Parts of our future work will involve the refinement of the type building approach. Although previous work already showed good results, the implementation of more type buildings and definition of city fabric types seems to be promising. We also want to integrate uncertainties in the type buildings, as no two buildings are alike. Implementing existing software tools for user behaviour is another approach to reduce the gap between simulation and the real buildings.

CONCLUSION

There are different options for generating dynamic simulation models on urban scale. All of them include manual effort for collecting data or generating the models. In this paper, we propose a methodology to generate dynamic building simulation models in Modelica from CityGML data sets. A self-developed Python tool *TEASER* extracts available geometric and semantic information (e.g. year of construction, function). The geometric processor supports the representation in LoD 1 and LoD 2. In a second step, *TEASER* enriches the collected information with statistical data based on elementary building parameters. This includes physical properties of wall construction as well as zoning the building and basic user behaviour. All information is used to generate a ready-to-use simulation model in the modelling language Modelica. The Modelica model is available open-source. Our approach enables to generate huge sets of dynamic simulation models, with a constant manual effort. The calculation time does not exceed a practical limit of time. Two use cases prove the concept. Use case 1 demonstrates the workflow with a generic and fictitious CityGML file. Use case 2 shows the capability of handling large data sets with a CityGML example file from Bad Godesberg, Germany. Both use cases show the concept of using CityGML as input for building simulation on urban scale.

We aim at further improvements to the presented methodology. Especially enhancing the type building approach, by including more building types and statistical data from various sources is promising. Furthermore, the implementation of dynamic user behaviour is a key factor. To this end, the authors observe international projects like IBPSA Project 1, Annex 66 and Annex 70 carefully. In addition, *TEASER* is open-source available to propagate the overall methodology (https://github.com/RWTH-EBC/TEASER).

ACKNOWLEDGEMENT

Grateful acknowledgement is made for financial support by BMWi (German Federal Ministry of Economic Affairs and Energy), promotional references 03ET1177A and 03ET1352A.

This work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 will develop and demonstrate new generation computational tools for building and community energy systems based on Modelica, Functional Mockup Interface and BIM standards.

REFERENCES

Becker, T., C. Nagel, and T. H. Kolbe. 2016. “CityGML - UtilityNetworkADE.” Accessed March 07, 2016. http://www.citygmlwiki.org/index.php/CityGML_UtilityNetworkADE.

Behnel, S: et. al. 2016. “lxml - XML and HTML with Python.” Accessed March 07, 2016. http://lxml.de/index.html.

Bezirksregierung Köln. 2016. “3D-Gebäudemodelle.” Accessed March 07, 2016. http://www.bezreg-koeln.nrw.de/brk_internet/geobasis/hoehenmodelle/3d_gebaeudemodelle/index.html.

Bigot, P. A. 2016. “PyXB: Python XML Schema Bindings.” Accessed March 07, 2016. http://pyxb.sourceforge.net/.

CityGML. 2016. “CityGML Schema download.” Accessed March 07, 2016. http://www.citygml.org/index.php?id=1540.

CityGML EnergyADE. 2016. “CityGML EnergyADE.” Accessed March 07, 2016. https://github.com/cstb/citygml-energy.

Deutsches Institut für Normung. *Energy efficiency of buildings - Part 10: Boundary conditions of use, climatic data*, no. DIN V 18599-10:2011-12.

Fuchs, M., J. Teichmann, M. Lauster, R. Streblow, and D. Müller. 2015. “Approach for simulation-based scenario analyses of district energy systems.” In *the 28th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*.

Gröger, G., T. H. Kolbe, C. Nagel, and K.-H. Häfele. 2012. “OGC City Geopgraphy Markup Language (CityGML) Encoding Standard.”.

Gröger, G., and L. Plümer. 2012. “CityGML – Interoperable semantic 3D city models.” *ISPRS Journal of Photogrammetry and Remote Sensing* 71: 12–33.

Hillebrand, G., G. Arends, R. Streblow, R. Madlener, and D. Müller. 2014. “Development and design of a retrofit matrix for office buildings.” *Energy and Buildings* 70: 516–22.

Kämpf, Jérôme H., and Darren Robinson. 2007. “A simplified thermal model to support analysis of urban resource flows.” *Energy and Buildings* 39 (4): 445–53. doi:10.1016/j.enbuild.2006.09.002.

Lauster, M., M.-A. Brüntjen, H. Leppmann, M. Fuchs, J. Teichmann, R. Streblow, C. van Treeck, and D. Müller. 2014a. “Improving a Low Order Building Model for Urban Scale Applications.” In *the 5th German-Austrian IBPSA Conference*, 511–18.

Lauster, M., P. Remmen, M. Fuchs, J. Teichmann, R. Streblow, and D. Müller. 2014b. “Modelling long-wave radiation heat exchange for thermal network building simulations at urban scale using Modelica.” In *the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*, 125–33. Linköping Electronic Conference Proceedings: Linköping University Electronic Press.

Lauster, M., J. Teichmann, M. Fuchs, R. Streblow, and D. Mueller. 2014c. “Low order thermal network models for dynamic simulations of buildings on city district scale.” *Building and Environment* 73: 223–31.

Lauster, Moritz, M. Fuchs, M. Huber, P. Remmen, R. Streblow, and D. Müller. 2015. “Adaptive Thermal Building Models and Methods for Scalable Simulations of Multiple Buildings Using Modelica.” In *the 14th International IBPSA Conference*.

Loga, T., N. Diefenbach, J. Knissel, and R. Born. 2005. *Entwicklung eines vereinfachten statistisch abgesicherten Verfahrens zur Erhebung von Gebäudedaten für die Erstellung des Energieprofils von Gebäuden*.

Mata, É., A. Sasic Kalagasidis, and F. Johnsson. 2014. “Building-stock aggregation through archetype buildings: France, Germany, Spain and the UK.” *Building and Environment* 81: 270–82.

Mayer, M. 2016. “Mako Templates for Python.” Accessed March 07, 2016. http://www.makotemplates.org/.

Miller, C., C. Hersberger, and M. Jones. 2013. “Automation of Common Building Energy Simulation Workflows Using Python.” In *the 13th International IBPSA Conference*.

Nouvel, R., K-H. Brassel, M. Bruse, E. Duminil, V. Coors, U. Eicker, and D. Robinson. 2015a. “SimStadt, a new Workflow-Driven Urban Energy Simulation Platform for CityGML City Models.” *International Conference Future Buildings and Districts - Sustainability from Nano to Urban Scale*.

Nouvel, R., R. Kaden, J-M. Bahu, J. Kämpf, P. Cipriano, M. Lauster, J. Benner, E. Munoz, O. Tournaire, and E. Casper. 2015b. “Geneses of the CityGML Energy ADE.” *International Conference Future Buildings and Districts - Sustainability from Nano to Urban Scale*.

O'Donnell, J., See, R. Rose, C., T. Maile, V. Bazjanac, and P. Haves. 2011. “SimModel: A Domain Data Model for Whole Building Energy Simulation.” In *the 12th International IBPSA Conference*, 382–89.

Python Software Foundation. 2016. “The ElementTree XML API.” Accessed March 07, 2016. https://docs.python.org/3.4/library/xml.etree.elementtree.html.

Reinhart, C. F., and C. Cerezo Davila. 2016. “Urban building energy modeling – A review of a nascent field.” *Building and Environment* 97: 196–202.

Robinson, D., F. Haldo, J. Kämpf, P. Leroux, D. Perez, A. Rasheed, and U. Wilke. 2009. “CITYSIM: Comprehensive Micro-Simulation of Resource Flows for Sustainable Urban Planning.” In *the 11th International IBPSA Conference*, 1083–90.

Roth, S. 2016. “Open Green Building XML Schema - gbXML.” Accessed March 07, 2016. http://www.gbxml.org/.

Schiefelbein, J., A. Javadi, M. Lauster, P. Remmen, R. Streblow, and D. Müller. 2015. “Development of a City Information Model to Support Data Management and Analysis of Building Energy Systems withing Complex City Districts.” *International Conference Future Buildings and Districts - Sustainability from Nano to Urban Scale*, 949–54.

Stadt Potsdam. “CityGML-Modell.” Accessed March 07, 2016. http://www.digitaler-gestaltplan-potsdam.de/docs/metadaten.html.

Swiss Society of Engineers and Architects. *Standard-Nutzungsbedingungen für die Energie- und Gebäudetechnik*, no. SIA Merkblatt 2024.

Wetter, M., M. Bonvini, and T. S. Nouidui. 2015. “Equation-based languages – A new paradigm for building energy modeling, simulation and optimization.” *Energy and Buildings.* doi:10.1016/j.enbuild.2015.10.017.