BIM2Modelica – An open source toolchain for generating and simulating thermal multi-zone building models by using structured data from BIM models

Christoph Nytsch-Geusen[1] Jörg Rädler[1] Matthis Thorade[2] Carles Ribas Tugores[3]

[1]Institut für Architektur und Städtebau, Berlin University of the Arts, Germany, nytsch@udk-berlin.de
[2]Modelon, Germany, matthis.thorade@modelon.com
[3]AEE INTEC, Austria, c.ribastugores@aee.at

Abstract

This contribution describes an open source toolchain which can transfer BIM models of 3D building constructions from CAAD programs into executable thermal multi-zone buildings models based on Modelica building energy simulation libraries. For this purpose, different open source libraries and tools were integrated into a Python-based software architecture of the toolchain: the IfcOpenShell/OCC libraries as the foundation for the import, analysis, and preparation of the BIM models; CoTeTo as the tool for the template-based code generation of the Modelica building models; the BuildingSystems library as the base for the thermal multi-zone building models; and JModelica as the simulation tool to perform the simulation analyses.

While the first part of the paper describes the general approach and the software architecture of the toolchain, the second part illustrates its application with an example of a real building.

Keywords: Building Information Modeling, IFC, Modelica code generation, Multi-zone building models

1 Introduction

The graphical modelling approach of Modelica, based on visualized components, connectors, and connections fits well for 2-dimensional topologies of energy plant systems, but not for 3-dimensional shapes of buildings and the topology of their constructions. On the one hand, the manual configuration of a thermal multi-zone building in Modelica in a graphical editor, based on components of a predefined library is an error-prone process. For example, the definition of a thirteen-zone building model in Modelica leads to a mo-file with more than 1,500 lines of code and a huge number of connect statements (Nytsch-Geusen, 2017). On the other hand, architects are using modelling tools such as ArchiCAD or Revit for their 3D building designs and often also Rhinoceros for prototypical designs. All these tools are able to export the geometry and the topology of a building design as a structured BIM model, normally in the IFC format.

For this reason, different research activities during the last years have been focused on the automatic generation of Modelica building models using IFC building models as the input (e.g. Thorade et al., 2015 and Reynders et al., 2017).

The toolchain described in Thorade et al. (2015) is based on the SimModel data model (O'Donnell et al. 2011). This data structure is able to store all relevant information for building energy simulation (the building construction and the related HVAC system), which is present in the BIM model itself (the IFC file) and which is optionally added by further data sources (e.g. data for missing material properties of building constructions). All these data are gained and combined by the use of the simulation tool Simergy (https://d-alchemy.com/products/simergy): with Simergy the user imports the architectural model as an IFC file, performs a space boundary analysis to obtain the topology information for the multi-zone building model, adds additional data with the Simergy GUI, and finally exports the entire data set as a SimModel file in the xml format. In the next step, a mapping tool takes the SimModel file, which instantiates and parameterizes the component and system models from present Modelica libraries, which reflect the problem of the BIM model. Because Simergy is a commercial simulation tool and the simulation tool used here is Dymola, not all of the toolchain is open source.

The approach of Reynders et al. (2017) describes a toolchain *Ifc2Modelica v0.2*, which is based on a Python framework. It can read IFC-files, determine the building topology for multi-zone building models, and generate Modelica building models in four different levels of complexity (LOC) for the Modelica IDEAS library (Jorissen et al., 2018). The model complexity reaches from a detailed thermal multi-zone building model, where each IFC entity is 1:1 mapped to a correspondent Modelica component model (LOC1) over some intermediate steps (LOC2, LOC3) down to a maximum simplified thermal single-zone building model, where all IFC entities are mapped to a small number of Modelica wall, window and door models (LOC4). The simplification from LOC1 to LOC4 takes place by

merging constructions of the same type and orientation and zones with similar conditions of use with the objective of a minimum loss of precision in the results and a maximum acceleration of the computation speed. The simulation analyses mentioned above were performed with the commercial *Dymola* tool, and the *Ifc2Modelica v0.2* toolchain is not released as an open source project.

The approach described in this paper demonstrates a Python-based complete open source toolchain, which reaches from the BIM modelling analysis up to the Modelica code generation and also supports an executable building simulation experiment, based on an open source Modelica simulation tool.

2 Toolchain

The BIM2Modelica toolchain from the IFC file up to the generated Modelica model includes three serial working Python modules (compare with Figure 1): a module for the BIM data import and analysis, a building data model for storing the analyzed and prepared information for building energy simulation, and the CoTeTo tool for generating thermal multi-zone building models based on the BuildingSystems library (http://www.modelica-buildingsystems.de).

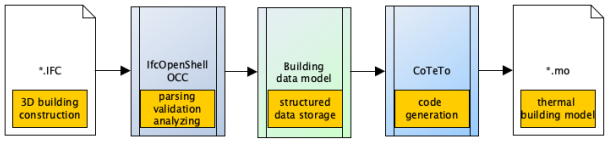

Figure 1. Software architecture of the BIM2Modelica toolchain.

2.1 BIM data analysis and preparation

The basis for the IFC data import and the subsequent data preparation is the IfcOpenShell library (IfcOpenShell, 2019) in combination with the OpenCascade library (pythonOCC, 2019). Based on these two Python libraries, a new library was implemented which enables the analysis and preparation of the imported IFC files in the following steps:

1. Filtering and sorting of all IFC types relevant for a thermal building model (types IfcSite, IfcSpace, IfcWall, IfcSlab, IfcDoor, IfCColumn, IfcWindow, etc.)
2. Extraction of all employed building constructions (type IfcMaterialLayerSet) from the imported IFC building model
3. Identification of the contact surfaces between the spaces (type IfcSpace) which represent the thermal zones and the adjacent building elements (types IfcWall, IfcSlab etc.) by means of a space boundary analysis (1st level space boundaries)
4. Determination of potential available openings in the building elements and the correspondent elements which fill them out (e.g. the relations between an IfcWall and an IfcWindow or IfcDoor)
5. Cutting of continuous building constructions (e.g. IfcWall or IfcSlab) which belong to more than one IfcSpace into sub components. Each of them can represent an individual thermal building element in the thermal building model with a potentially different thermal boundary condition (2st level space boundaries).

2.2 Building data model

The building data model consists of a data structure which stores all of the information in an intermediate step, before it is used for the code generation of the thermal building model, expressed in Modelica. The building data model is realized by a couple of Python classes which are able to store all of the required geometry and topology information of each thermal zone and individual building element. It also includes a list of all of the construction types used. Further, information regarding the employed building materials, the type of use for each thermal zone (ventilation rates, internal heat sources, set temperatures for heating and cooling), the building orientation and the building location can be added, if not already present in the IFC file.

The information collection of the building data model covers the typical amount of data for the parametrization of multi-zone thermal building models. Up to now, it has exclusively been used as a database for Modelica code generation, but in principle, it could also be applied to the creation of multi-zone building models for other simulation tools such as EnergyPlus or TRNSYS.

2.3 Modelica code generation

In the next step of the toolchain, the Modelica building models are generated using the information stored in the building data model. For this purpose, the Python based module CoTeTo (**Co**de **Te**mplating **To**ol) is used, which was developed in the EnEff-BIM project (see Thorade et al., 2015).

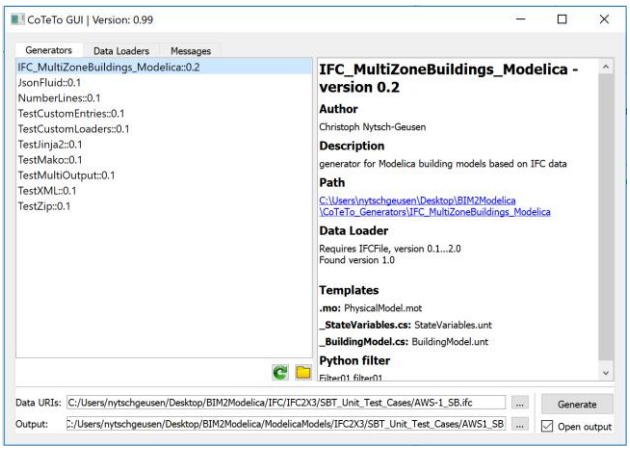

Figure 2. GUI of the CoTeTo code generation tool.

DOI 10.3384/ecp1915733

CoTeTo can be flexibly configured with pluggable input, filter, and output components which support the single steps of data acquisition, preprocessing and code generation using a template system. It can be used standalone with a GUI (compare with Figure 2) or as an imported module in Python applications. The code generation step in CoTeTo is based on the Mako template engine (Mako, 2019).

2.4 BuildingSystems Library

The CoTeTo code generator for thermal multi-zone models code was designed for the predefined model classes (thermal zones, walls, windows, doors etc.) of the Modelica BuildingSystems library (Nytsch-Geusen et al., 2016). As other Modelica libraries for building energy simulation such as IDEAS, AIXLib and Buildings, the BuildingSystems library uses as its core the same Modelica IBPSA library (Modelica IBPSA library, 2019), which is the successor of the former Annex 60 library (Wetter et a. 2015).
The following code excerpt demonstrates the principle, upon which the model classes of the BuildingSystems library are instantiated and parameterized during the code generation process, based on a Mako template. Access to the required building information stored in the building data model takes place in the example over the Python dictionary `data`. Outgoing from a generalized template definition in Mako

```
% for ele in data['elementsOpaque']:
BuildingSystems.Buildings.Constructions.Walls.WallThermal1DNodes ${ele.name}(
% if generatorCfg['MODELICA_SWITCHES'].getboolean('surTemOut'):
  show_TSur = true,
% endif
  redeclare ${ele.constructionData} constructionData,
  angleDegAzi = ${ele.angleDegAzi},
  angleDegTil = ${ele.angleDegTil},
  AInnSur = ${ele.AInnSur},
  height = ${ele.height},
  width = ${ele.width});
% endfor
```

the Modelica code for a flexible number of wall models of a thermal building model can be generated:

```
BuildingSystems.Buildings.Constructions.Walls.WallThermal1DNodes wall_4(
 redeclare Construction1 constructionData,
    angleDegAzi = 90.0,
    angleDegTil = 90.0,
    AInnSur = 0.0,
    height = 7.8,
    width = 10.000000000000002);


BuildingSystems.Buildings.Constructions.Walls.WallThermal1DNodes wall_5(
 redeclare Construction1 constructionData,
 angleDegAzi = 0.0,
 angleDegTil = 90.0,
 AInnSur = 0.0,
 height = 7.8,
 width = 9.849999999999998);
...
```

The syntax of the Mako language is similar to the Python language, but it works with the `%` sign before control statements and without indents. Therefore, a for-loop or a conditional statement in Mako needs an **% endfor** and an **% endif** in addition to the `% for` and the `% if`. Expressions within curly braces, e.g. `${ele.name}`, are evaluated, and the result is used for the code generation process. The example of the Mako template also illustrates the flexibility of the code generation process. If the Boolean expression

```
generatorCfg['MODELICA_SWITCHES'].getboolean('surTemOut'):
```

becomes true, the Modelica parameter `show_TSur = true` is generated in the Modelica code; otherwise it is not.

2.5 Toolchain validation

The toolchain was tested with a set of 33 BIM building models in the IFC 2x3 format, which cover a broad spectrum of possible geometrical and topological structures of building constructions (Bazjanac, 2017). Figure 3 shows a subset of these building models.

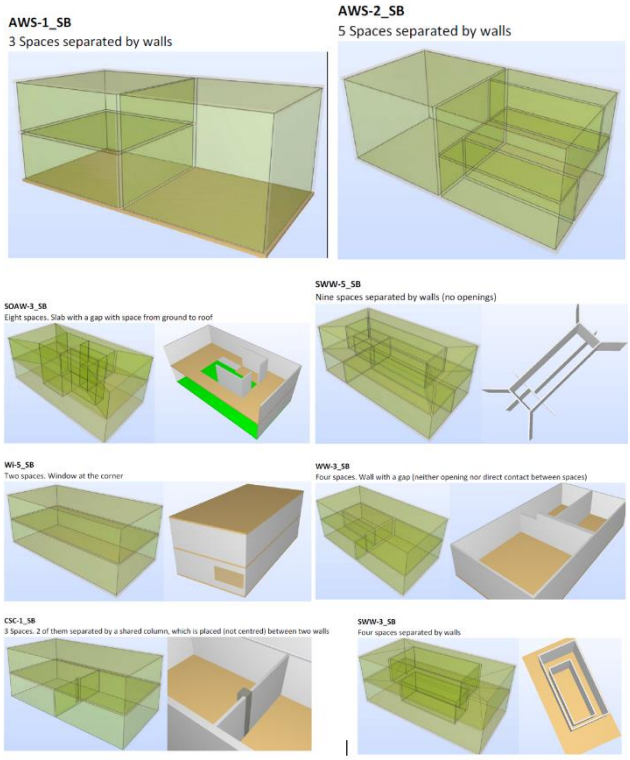

Figure 3. Exemplary IFC test cases for validating the entire toolchain from the BIM data import through the data preparation to the Modelica code generation.

In addition, three further IFC2X3 models with one, two and thirteen thermal zones were used for the validation

procedure. All of the building models were constructed in ArchiCAD and afterwards exported as (IFC) BIM-models.

The validation process was executed for all of the models in three serial steps:

1. Correct translation of the BIM model into the building data model.
2. Correct generation of the Modelica building model with the Mako template.
3. Successful simulation of the generated building models with JModelica and Dymola and achieving the same simulation results with both tools.

With the help of these test cases, many potential failures and weak spots in the algorithms of the IFC import and data preparation module (e.g. incorrectly calculated geometries and topologies), the building data model (e.g. missing attributes) and the code generation template used in CoTeTo (e.g. required additional features for a more flexible code generation) could be detected, fixed, and improved.

2.6 Simulation experiment with JModelica

After the multi-zone building model code was generated with CoTeTo, a simulation experiment could be performed with a Modelica tool. Because the objective of the development of the toolchain was a pure open source solution, JModelica (http://JModelica.org) was used for this purpose. The definition of a Modelica simulation experiment in JModelica takes place in Python script, in which the three steps model translation, model simulation and, result visualization have to be defined. The JModelica compiler (Python module pymodelica) obtains the Modelica model over the method `compile_fmu()` and generates an executable FMU in version 1.0 or 2.0.

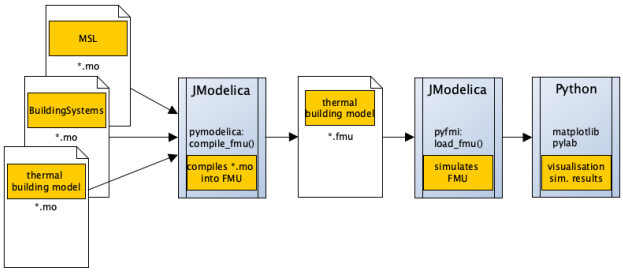

Figure 4. Compilation and simulation of the thermal building models with JModelica.

In the following step this FMU is taken by the JModelica run time system (Python module pyfmi) by using `mymodel`, an instance of the Python class `FMUModelBase`. This instance is generated as the return value of the function call `load_fmu()`. The two methods calls `myModel.simulate_options()` and `myModel.simulate()` configure the numerical options and start the simulation experiment. The simulation results are stored in a Python dictionary and visualized with a suitable graphical Python library such as matplotlib or pylab after the simulation experiment is performed (compare with Figure 4 and the following excerpt of a Python script, which defines the simulation experiment):

```
# compile model to fmu
from pymodelica import compile_fmu
fmu = compile_fmu('MultiZoneBuilding',…)

# load the fmu
from pyfmi import load_fmu
myModel = load_fmu(fmu)

# simulate the fmu and store results
opts = myModel.simulate_options()
opts['solver'] = "CVode"
opts['ncp'] = 240
res = myModel.simulate(start_time=0.0,
final_time=864000, options=opts)

# plotting of the results
import pylab as P
fig = P.figure(1)
y1 = res['ambient.TAirRef']
y2 = res['building.TAir[1]']
y3 = res['building.TAir[5]']
y4 = res['building.TAir[12]']
t = res['time']
P.subplot(2,1,1)
P.plot(t,y1,t,y2,t,y3,t,y4)
P.legend(['ambient.TAirRef','building.TAir
[1]'],…)
P.ylabel('Temperature (K)')
P.xlabel('Time (s)')
P.show()
```

3 Case study

The described approach of the toolchain was evaluated by the example of a small residential living unit, the Rooftop building, which was developed for the Solar Decathlon Europe 2014 (SDE 2014) in Versailles, France (http://www.solardecathlon2014.fr/en/) by a student team from UdK Berlin and TU Berlin (see Figure 5).

Figure 5. The realized prototype of the Rooftop building on the SDE 2014 competition site in Versailles, France.

This rooftop construction was designed as a solar plus energy living unit, which can be placed on top of the building stock (compare with Figure 6) and can be air-

DOI 10.3384/ecp1915733

conditioned and supplied by its own gained energy all the year around. A detailed description of the Rooftop building incl. the technologies used (reversible heat pump, adaptable photovoltaic facades, thermal and electrical storage management etc.) can be found in Team Rooftop (2014).

Figure 6. The concept of the Rooftop building as a solar living unit for the building stock for dense city districts.

The Rooftop building was modelled for the case study as a 3D BIM model in ArchiCAD. Starting from this information base, an IFC2X3 model was exported. Figure 7 shows the visualization of this IFC model with all construction elements, and Figure 8 shows only the part of the building model which is relevant to a building energy simulation.

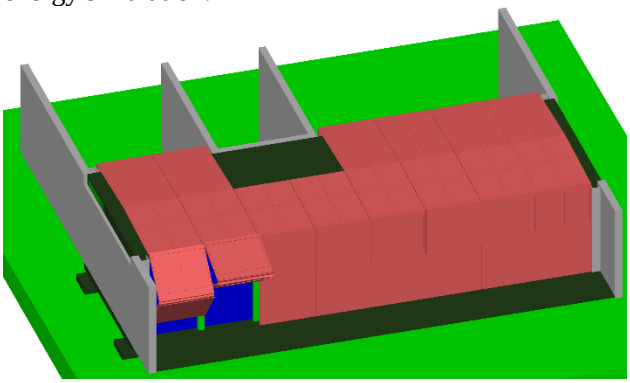

Figure 7. BIM model Rooftop building with all building elements.

Figure 8. BIM model of the Rooftop building, reduced to the relevant building elements for building energy simulation.

The inner building structure includes four thermal zones, two of which can be air-conditioned by a floor heating and a cooling ceiling system (zones lab and seminar). The other two are only thermal buffer zones with free floating temperatures (zones toilet and core). The building construction consists of wooden lightweight building elements in combination with large glass facades.

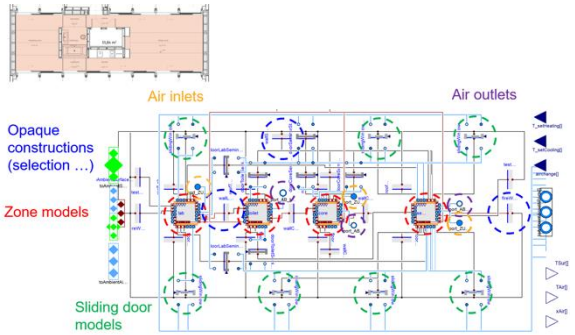

Figure 9. Generated Modelica building model.

At present, the Modelica code generation is restricted to the thermal building model (see Figure 9); the HVAC system of the building energy system still has to be configured by hand.

The generated Modelica model of the Rooftop building was simulated with JModelica for a period of four hot summer days for the location Berlin. In Figure 10, the outside air temperature and the free-floating air temperatures of the four thermal zones are illustrated. Because the air change of the generated building model is suppressed and the large transparent facades are unshaded in the configuration of the simulation experiment, the air temperatures in the seminar zone and the lab zone show the typical increasing overheating behavior of a “glass house” over the time.

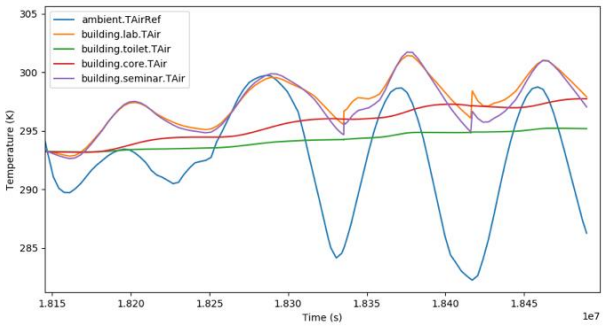

Figure 10. Simulated indoor climate of the Rooftop building during four warm summer days (location Berlin).

4 Summary and Outlook

A Python-based open source toolchain for generating thermal multi-zone building models from BIM models for the Modelica BuildingSystems library was successfully implemented, validated, and evaluated by

means of the case study of the Rooftop building. Up to now, the code generation process is limited to the building construction; the HVAC system of the building has to be manually added.

The source code of the BIM2Modelica toolchain incl. the code generation tool CoTeTo, the Modelica BuildingSystems library, and a set of BIM test cases is available for free and can be downloaded as one software package from GitHub (https://github.com/UdK-VPT/BIM2Modelica).

The future development of the toolchain will take place in collaboration with research partners within the IBPSA project 1 (https://ibpsa.github.io/project1) in work package 2.2 “Building Information Modeling”.

Future developments of the BIM2Modelica toolchain will focus on automatic reduction of the building model complexity dependent on the given boundary conditions (orientation of façade elements, conditions of use for the zones), similar as described in Reynders et al. (2017).

Further, additional specialized CoTeTo templates for C# code generation that supports a building model visualization for Unity (https://unity3d.com/de) are under development (see also Nytsch-Geusen et al., 2017).

Acknowledgements

The research described in this paper was conducted within the research project “EnEff BIM: Planung, Auslegung und Betriebsoptimierung von energieeffizienten Neu- und Bestandsbauten durch Modellierung und Simulation auf Basis von Bauwerkinformations-modellen” and funded by the Federal Ministry for Economic Affairs and Energy in Germany (reference: 03ET1177D).

References

Bazjanac, V. (2017). Testing space boundaries that transcribe complex CAD building geometry into surface geometry usable by EnergyPlus and similar building energy performance simulation engines. Internal report, UdK Berlin, Germany.

IfcOpenShell (2019). The open source IFC toolkit and geometry engine - http://ifcopenshell.org/python.html (last access 2019 Jan 21)

Jorissen, F, Reynders, R.; Baetens, R.; Picard, D.; Saelens, D. and Helsen, L. (2018). Implementation and Verification of the IDEAS Building Energy Simulation Library. Journal of Building Performance Simulation, 11 (6), 669-688, doi: 10.1080/19401493.2018.1428361.

Mako (2019). Mako templates for Python - http://www.makotemplates.org (last access 2019 Jan 21)

Modelica IBPSA library (2019) - https://github.com/ibpsa/modelica-ibpsa (last access 2019 Jan 21).

Nytsch-Geusen, C.; Banhardt, C.; Inderfurth. A.; Mucha, K.Möckel, Jens; R., Jörg; Thorade, M.; Tugores, C. (2016). BuildingSystems – Eine modular hierarchische Modell-Bibliothek zur energetischen Gebäude- und Anlagensimulation. BAUSIM 2016 IBPSA. Conference Proceedings, Dresden, Germany.

Nytsch-Geusen, C.; Inderfurth, A.; Kaul, W.; Mucha, K.; Rädler, J.; Thorade, M. and Tugores, C.R. (2017). Template based code generation of Modelica building energy simulation models. 12th International Modelica Conference, Conference Proceedings, Prag, Czechia.

O’Donnell, J.; See, R.; Rose, C.; Maile, T., Bazjanac, V. and Haves, P. (2011). SimModel: A domain data model for whole building energy simulation. In Proceedings of the 12th IBPSA Building Simulation Conference, Sydney, Australia.

pythonOCC (2019). pythonOCC – 3D CAD for python - http://www.pythonocc.org (last access 2019 Jan 21).

Reynders, G.; Andriamamonjy, A.; Klein, R; Saelens, D. 2017 Towards an IFC-Modelica tool facilitating model complexity selection for building energy simulation (2017). 15th IBPSA Building Simulation Conference, Conference Proceedings, San Francisco, USA.

Team Rooftop (2014), Deliverable 6 & 7 of the Solar Decathlon Europe 2014. Official documentation of the Rooftop project. UdK Berlin and TU Berlin, Germany.

Thorade, M.; Rädler, J.; Remmen, P.; Maile, T.; Wimmer, R.; Cao, J; Lauster, M.; Nytsch-Geusen, C.; Müller, D. and van Treeck, C. (2015) An open toolchain for generating Modelica code from Building Information Models. 11th International Modelica Conference, Conference Proceedings, Versailles, France.

Wetter, M.; Fuchs, M.; Grozman, P.; Helsen, L., Jorissen, F.; Lauster, M.; Müller, D.; Nytsch-Geusen, C.; Picard, D.; Sahlin, P.; and Thorade, M. (2015). IEA EBC Annex 60 Modelica Library - An international collaboration to develop a free open-source model library for buildings and community energy systems. 14th IBPSA Building Simulation Conference, Conference Proceedings, Hyderabad, India.