MODEL TRANSFORMATION FROM SIMMODEL TO MODELICA FOR BUILDING ENERGY PERFORMANCE SIMULATION

Jun Cao[1], Tobias Maile[1], James O'Donnell[2], Reinhard Wimmer[1], and Christoph van Treeck[1]
[1]Institute of Energy Efficient Building E3D, RWTH Aachen, Germany
[2]School of Mechanical and Materials Engineering and Electricity Research Centre, University College Dublin, Ireland
{cao, maile, wimmer, treeck}@e3d.rwth-aachen.de, james.odonnell@ucd.ie

ABSTRACT

This paper demonstrates a model transformation tool between the Building Information Model (BIM) and Modelica schemas for Building Energy Performance Simulation (BEPS) purposes. Automated reuse of data in BIMs to accelerate BEPS model development is now a promising approach for engineers. However, using BEPS tools such as Modelica to generate building simulation models is currently difficult and time consuming due to the largely manual data input. The ability to import data of a BIM into Modelica-based BEPS tool would improve this process significantly.

In order to address the challenge, this paper proposes a model transformation prototype to convert Industry Foundation Classes (IFC) based BIMs into object-oriented Modelica simulation models. The proposed approach uses SimModel data model as a placeholder for IFC. SimModel currently contains richer HVAC data definitions than IFC for the purposes of BEPS and can be easily extended to store the additional data required by Modelica. This prototype demonstrates the potential to reduce development time of Modelica models by reusing building information data stored in BIMs.

INTRODUCTION

In traditional building simulation programs, e.g., DOE-2, ESP-r, and EnergyPlus, component models frequently integrate their own numerical solver and mix program flow logic with equations that simulate physical behavior (Wetter et al., 2014, 2011; Wetter, 2009). This leads to a program code that is difficult to use for the design and analysis of building energy and control systems. On the other hand, equation-based object-oriented Modelica modeling is separating the numerics from the engineering equations. Thus, the equations are more transparent and the user has more flexibility to change component behavior. This can effectively speed up the process of developing innovative building energy models, since the user can easily customize components.

Traditional manual BEPS model development often takes significant effort and can result in numerous errors and omissions, and inevitably adds dramatically to the cost of the project (Bazjanac and Maile, 2004; Bazjanac and Crawley, 1997). A promising solution to the problem proposes linking BIM to Modelica (Yan et al., 2013; Jeong et al., 2014; Wimmer et al., 2014; Basarkar et al., 2012; Aksamija, 2012) in a manner that allows designers, e.g., control engineers, simulation specialists, energy consultants, etc., to create Modelica models by reusing building information data stored in BIMs.

Yan et al. (2013) and Jeong et al. (2014) present the most relevant works in model conversion from BIM to Modelica. However, their work focuses on transforming geometry for thermal and daylighting modeling based on the Lawrence Berkeley National Laboratory (LBNL)'s Modelica Buildings library (Wetter, 2009). Their methodology, using Auto Revit APIs, converts building models defined in a proprietory CAD format.

In this paper, we focus on converting HVAC systems from IFC-based BIMs into Modelica, in order to accelerate BEPS model development. The targeted Modelica BEPS library AixLib (EBC, 2014; Constantin et al., 2014) was developed specifically for the simulation of HVAC systems of buildings. Other libraries such as LBNL's Buildings (Wetter, 2009) are focusing on HVAC systems of the building sector as well.

We will use SimModel (O'Donnell, 2011) as a placeholder for IFC in our development. Because IFC is the open standard for BIM but does not contain adequate HVAC data definitions for BEPS. SimModel is the only BIM format that contains the necessary data for BEPS and the structure of this data model aligns with the structure of IFC. In addition, geometry definitions contained in IFC can be imported into SimModel and HVAC definitions added. This is the starting point for our work. The outcome is a prototype that transforms SimModel into Modelica data.

The model transformation from SimModel to Modelica presents a unique challenge that must account for significant differences in their respective model structures. For example, the model component and parameter definitions, the modeling hierarchy and level of detail (LOD) of SimModel and Modelica differ substantially.

Due to different naming conventions and data representations, the semantic definitions of parameters and components in these two models differ. For instance, the boiler efficiency curve of SimModel is defined by profile functions, and is represented with fixed values in a matrix in Modelica. Moreover, there is a hierarchy of building and HVAC components stored in SimModel but Modelica maintains its structure by defining explicit physical connections between model compo-

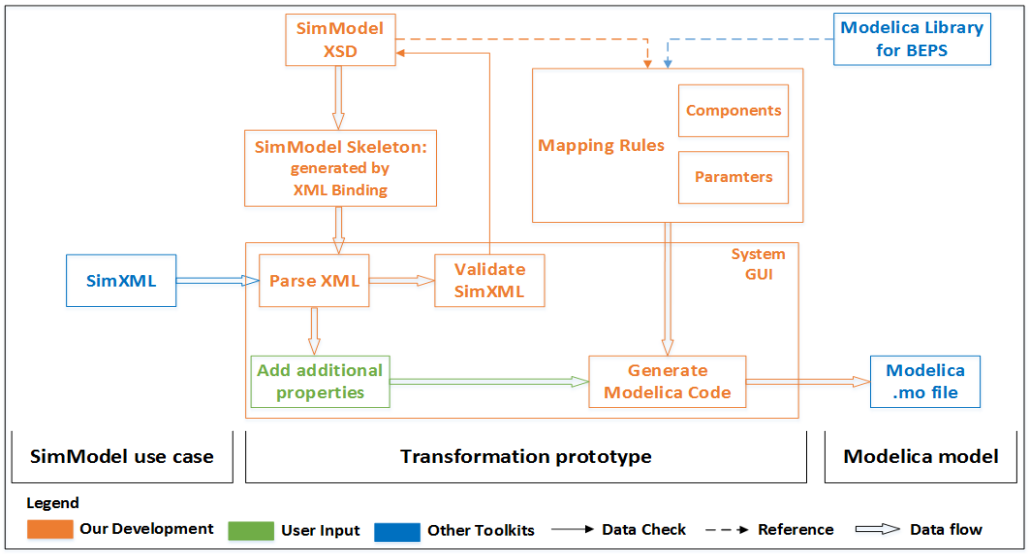

Figure 1: System pipeline for model transformation. The SimModel use case is stored as a SimXML file (left), the transformed Modelica model is stored in the .mo file (right) as an output of the system GUI (middle).

nents and containment relationships. Currently, SimModel only entails the predefined LOD originating in EnergyPlus, on the contrary, Modelica allows the definition of deeper LODs.

The missing hierarchical structure on Modelica side is not a major problem for SimModel transformation, since the conversion is from a hierarchy structure to a more flexible model representation. Furthermore, the increased LOD of Modelica requires the ability for the user to add additional data. In contrast, the semantic mismatches on model definitions are more problematic, Wimmer et al. (2014) define parameter and component mapping rules between SimModel and Modelica that are integrated into this prototype.

In this paper, the System Overview section illustrates the system processing pipeline of model transformation and Technical Specifications provides a detailed description of the system prototype. In order to demonstrate the entire pipeline, the Experiment section illustrates an automatically generated Modelica model transformed from the first SimModel use case. Conclusions and future work are discussed in the final sections.

SYSTEM OVERVIEW

In order to address the fundamental differences (LOD, hierarchy, schematic mismatches) between SimModel and Modelica data models, the prototype of our system, shown in Figure 1, is developed for simulation specialists to bridge these gaps with the following stages:

A given SimModel use case of specific HVAC systems and its building geometry are stored in an XML-based data file SimXML. This SimXML file, generated by Simergy (LBNL, 2014), is the data source of the transformation prototype application.

The data model stored in SimXML file will be loaded by a graphical user interface (GUI) to validate its XML syntax according to the XML Schema Definition (XSD) of SimModel. If no error is found in SimXML, the model hierarchy of SimXML data will be parsed and converted into C++ class instances, in memory, by calling APIs generated in the SimModel skeleton class.

Subsequently the user can input additional simulation parameter values through the GUI for specified HVAC properties. Afterwards, a Modelica code generator embedded in the GUI generates Modelica model based on the Mapping Rules and output a .mo file to save the Modelica code (Figure 1, Transformation prototype).

In the background of the system GUI, a multi-layer XML Binding is developed based on an open source XSD binding parser CodeSythesis XSD (CodeSynthesis, 2011) to generate the C++ skeleton class of SimModel by parsing SimModel XSD. The Mapping Rules illustrated in the background is a data container designed for saving mapping rules (Wimmer et al., 2014) between SimModel and Modelica objects. In this prototype, we focus on a specific Modelica buildings library AixLib (EBC, 2014; Constantin et al., 2014) and develop mapping rules that define object and property mappings for HVAC use cases.

TECHNICAL SPECIFICATIONS

Technical specifications of the model transformation system are presented in following sub-sections:

- An overview of SimModel hierarchy, including the XML binding framework generated for parsing SimXML.
- Geometry and HVAC representations of SimModel and AixLib are compared to illustrate model differences.
- Different data mapping rules are defined to convert SimModel into Modelica.

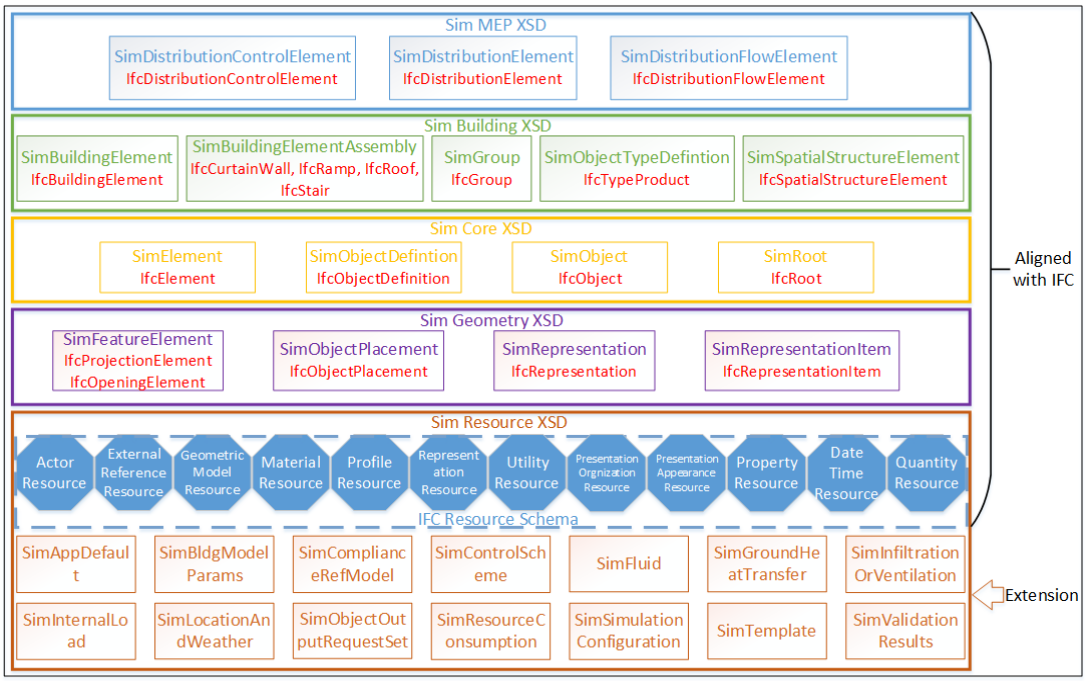

Figure 2: SimModel hierarchical structure (designed for BEPS) includes five sub-schema namespaces: Sim Resource, Sim Geometry, Sim Core, Sim Building and Sim MEP.

- Lastly, the data structure of Modelica model is analyzed to support the transformation triggered by data mapping rules.

SimModel Hierarchy

SimModel hierarchical structure contains five sub-schema namespaces to store energy performance-related data, e.g., building geometry, HVAC system, physical property data assigned to the HVAC system and simulation configuration. Figure 2 illustrates parent classes of each sub-schema namespace displayed. The corresponding IFC classes are also marked in Figure 2 by red text.

Sim Resource: located in the bottom layer of the SimModel hierarchy, represents the resource data assigned to building elements or HVAC systems. Besides basic IFC functionalities, illustrated by blue octagons in figure 2, additional functions maintaining energy-performance data are extended by SimModel to support BEPS. For example, the object SimTemplate is designed as an object container storing various library entries into a reusable template. E.g., the construction template contains all the different material layer sets for different building elements as defined by a given standard.

Sim Geometry: this geometry schema provides multiple-representation of geometry shapes defined for rendering the building envelope or the other inner structures, e.g., building site, building storey, ceiling, etc.

Sim Core: basic data types and model structures of SimModel objects are defined in this core schema. For example, the object id, object model name and its model type are defined in SimObject class of this schema.

Sim Building: high-level building element structures, e.g., building site, storey, reference to the low-level geometric shapes, are defined in this schema. Based on these high-level building elements, different thermal zone definitions and their HVAC group information are stored in the SimGroup class of this sub-schema.

Sim MEP: mechanical, electrical and plumbing elements of HVAC systems are defined in this sub-schema. As used in our use case example described later in this paper, a hot water boiler, a water pump and a heat radiator are defined by Sim MEP schema.

XML Data Binding Framework

XML data binding is the process of extracting data from a structure representation of XML documents and presenting it as a hierarchy of objects that correspond to a document vocabulary. This allows data-centric applications to manipulate XML data in a way that is more natural than using the Document Object Model (DOM).

Using the SimModel XSD (XML instance specification), we generate C++ classes that:

- Reflect the tree structure of the XML schema as in-memory object.
- Access the data stored in the tree efficiently, e.g., SAX (Simple API for XML)-like mapping.

In general, approaches for automated mapping of XML Schemata to C++ are broadly classified into two categories (CodeSynthesis, 2011): 1) in-memory tree mapping and 2) stream-oriented parsing. The tree mapping represents the information stored in XML

documents as a tree-like data structure suitable for in-memory processing. Stream-oriented paring is a SAX-like mapping which represents the data stored in XML as a hierarchy of vocabulary-specific parsing events. Table 1 summarizes key advantages of these two C++ binding methods:

Table 1:
Binding method comparison between tree mapping and stream-oriented parsing (CodeSynthesis, 2011)

TREE MAPPING	**STREAM-ORIENTED**
Ready to use default data type mapping system for mapping XML schema types to suitable C++ types	Support customized type mapping system to construct your own in-memory data representation
Complete XML document view and referential integrity	Perform immediate processing as parts of the XML document become available
Optional association with underlying DOM nodes	Handle XML documents that are too large to fit into memory
Support serialization back to DOM or XML, ID/IDREF cross-referencing, etc.	Small footprint, including code size and runtime memory consumption

Based on the pros and cons of XML binding methods described above, we developed a multi-layer XML binding parser that combines advantages of these two bindings to generate C++ skeleton classes of SimModel. As a result, the generated C++ SimModel skeleton represents the given vocabulary of a SimXML as well as parsing and serialization code. The new developed parser will generate a complete structure view of a SimXML file based on the tree mapping. Then parsing only parts of the SimXML file into memory objects via the stream-oriented binding for handling SimXML files that are too large to fit into memory.

The structure of the multi-layer XML binding parser developed for SimModel is illustrated in Figure 3:

Underlying XML Parser: this is a system specified XML parser library which makes it easy to give our binding skeleton the ability to read and write XML data. A shared library is provided for parsing, generating, manipulating, and validating XML documents using the DOM, SAX, or SAX2 APIs.

Type Mapping: this layer is used to define type mapping between XML schema and C++ types. The XML binding compiler, e.g., CodeSythesis XSD, XMLSpy, has a set of predefined mapping rules that map built-in XML schema types to suitable C++ types. By providing customized type maps user can override these predefined rules. For instance, map the pump speed defined in SimModel (see Figure 3) into a C++ enum type pumpSpeed of two variables, i.e., ConstantSpeed and VariableSpeed.

Generate Skeleton Class: the C++ skeleton class of SimModel is generated by calling a user specified XML data binding complier. In our system, we select the open-source, cross-platform W3C XML schema binding compiler CodeSythesis XSD as the default SimModel XSD binding parser in order to provide complete control of the skeleton class generated.

Override Virtual Functions in the Skeleton Class: based on the generated skeleton class, the system GUI can then access the data stored in a SimXML file by overriding virtual functions that semantically correspond to SimModel elements rather than dealing with the intricacies of reading and writing XML.

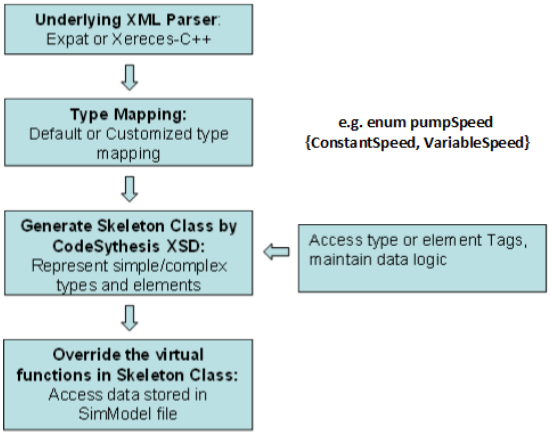

Figure 3: The multi-layer XML binding parser for mapping SimModel XSD into C++ skeleton class

Comparison of Geometric Representation between SimModel and AixLib

In this section, we illustrate model differences in geometry representations between SimModel and AixLib (EBC, 2014; Constantin et al., 2014).

In the upper part of Figure 4, geometric hierarchy of a single zone building office defined in SimModel is decomposed into a top-down tree structure. The geometric representation of SimModel fully complies with the spatial structure decomposition defined by the IFC Implementation Guide (Liebich, 2009). As sim classes displayed in Figure 4, the geometric building envelope is constructed by linking a set of geometry elements, such as link SimSite with SimBuilding, SimBuilding with SimBuildingStory, SimBuildingStory with SimWindow, SimSlab and SimWall.

In contrast to the geometric hierarchy of SimModel, there is no hierarchical geometry representation in AixLib. The lower part of Figure 4 shows the transformed building envelope defined in Modelica. The resistor-capacitor (RC) models of outerwall (the wall has a window) and innerwall (combined all internal adiabatic walls in this case) are generated according to the SimWall objects of SimModel. Afterwards, the RC models of outerwall and innerwall are linked to the thermal zone.

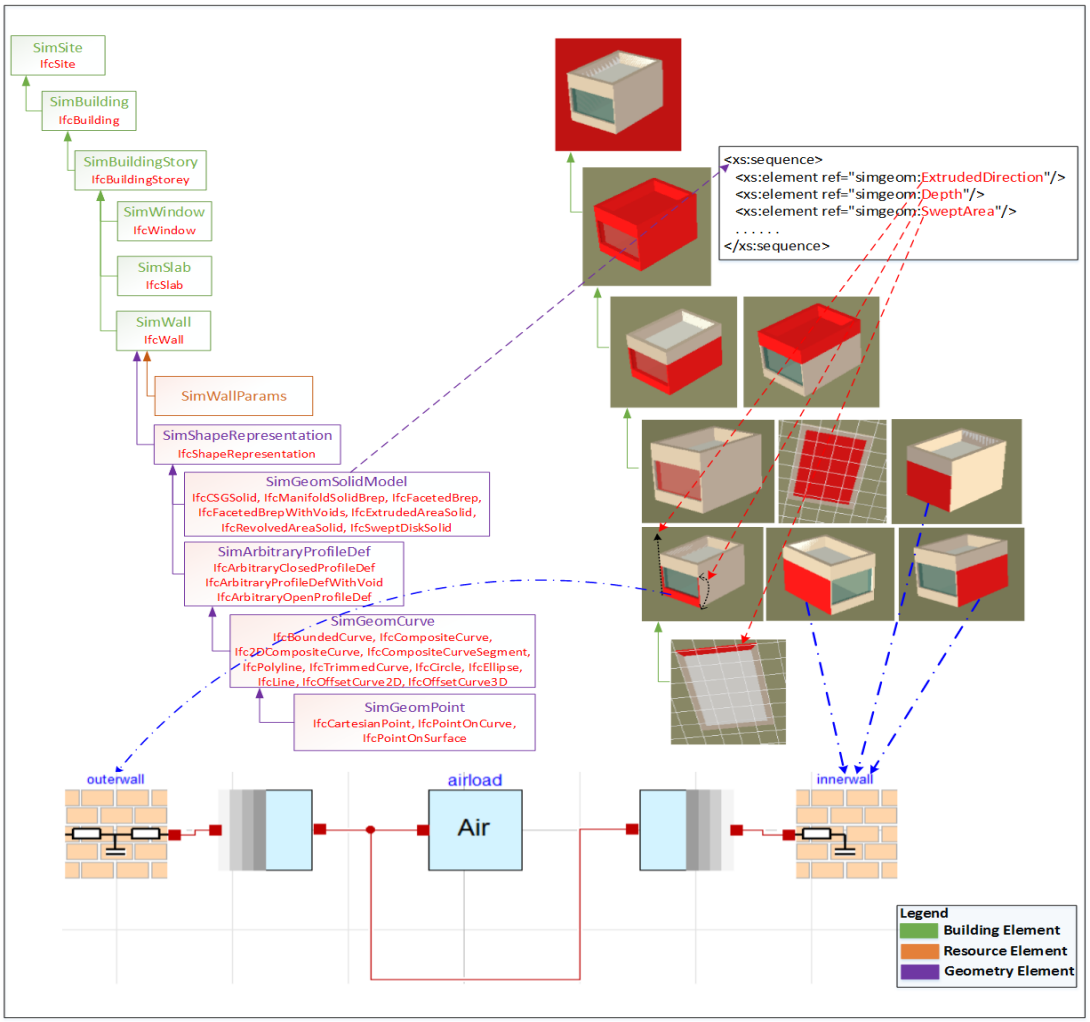

Figure 4: Geometric comparison between SimModel and AixLib. As illustrated by blue arrows, geometric hierarchy of SimModel wall elements will be transformed into Modelica RC models.

Comparison of HVAC Representation between SimModel and AixLib

Figure 5 shows a hot water looping system defined in SimModel and AixLib respectively.

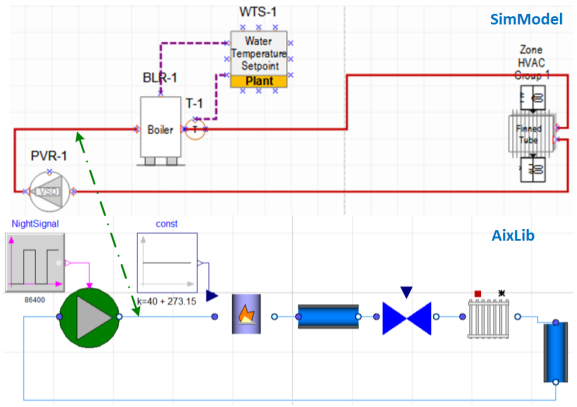

Figure 5: HVAC Comparison between SimModel and AixLib. The use case is visualized in Simergy (upper) and Dymola (lower).

Generally, the following differences occur between HVAC systems defined in SimModel and Modelica-based BEPS libraries, e.g., AixLib:

1. HVAC Component and Parameter Mapping: normally a HVAC component or parameter defined in SimModel will be transformed into a corresponding component or parameter defined in Modelica, e.g., in Figure 5, the hot water boiler of SimModel is mapped to the same type of boiler in Modelica. Nevertheless, some HVAC components or parameters have to be mapped into multiple Modelica components or parameters. For example, the SimModel radiator in Figure 5 has to be transformed into a radiator and an additional valve as a radiator is controlled separately in Modelica.
2. Looping Flow Direction: SimModel supports directional looping flow in a predefined system configuration. Modelica can support bi-directional flows in a flexible configuration. As shown in Figure 5, each HVAC component of SimModel has an inlet and outlet port, e.g., the red triangles on the boiler and radiator.
3. Different Model LOD: currently, SimModel only

supports predefined LOD originating in EnergyPlus, on the contrary, Modelica allows to define deeper LODs based on simulation requirements. As shown in Figure 6, the detailed model structure of boiler is defined in the 2nd model level. More detailed models of internal components of the 2nd level can be defined in deeper LODs, such as in 3rd and 4th levels.

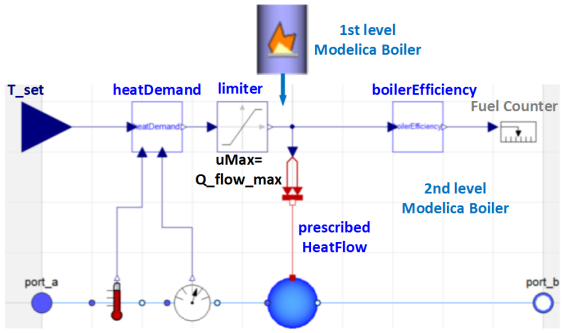

Figure 6: Deeper LODs of Modelica Model

Mapping rules from SimModel to Modelica

Based on the comparison between SimModel and AixLib, 6 different mapping rules are defined to convert SimModel into Modelica:

1. One to One Mapping: a single parameter or component of SimModel will be mapped into another parameter or component of Modelica.
2. Many to One Mapping: multiple SimModel parameters or components will be converted into a single Modelica parameter or component.
3. One to Many Mapping: a single SimModel parameter or component will be converted into multiple Modelica parameters or components.
4. Gap: additional data needs to be added into SimModel in order to represent the Modelica parameter or component.
5. Transformation: a parameter or component in SimModel will be transformed into a different representation in Modelica. For example, the boundary conditions of HVAC components are normally represented by matrix-based data instances in SimModel, and will be defined by functions in Modelica.
6. Combination: it is a combined mapping from rule 1 to 5. Different combinations of these rules are possible. For example, a matrix-based data pairs in Modelica represent the part-load factor and its corresponding energy efficiency of a boiler. We need to map the maximum and minimum values of the part-load ratio defined in SimModel (2 parameters) into the upper and lower bounds of the single Modelica parameter part-load factor based on rule 2, and interpolate the other value instances of the part-load factor in between the upper and lower bounds. Then substitute these interpolated factor values into a profile function of SimModel to calculate their corresponding boiler efficiencies based on rule 5. Afterwards, the efficiency values will be saved into the matrix-based data pairs.

These mapping rules, mainly driven by a set of use cases, will be a first step to illustrate the model transformation between SimModel and Modelica. The definition of these rules is described in Wimmer et al. (2014).

Modelica Code Structure

In this section, we introduce the code structure of BEPS models generated into a Modelica .mo file. The following two parts are normally defined in the Modelica .mo file to describe BEPS models:

- Model Component Initialization: in this section model objects are initialized, e.g., HVAC components pre-defined in the building library, with specific parameter values. For example, the following code is to initialize a boiler of AixLib with the maximum heat output 1300W fluid volume (inside the heat generation unit) starts at 328.15K: **AixLib.HeatGeneration.Boiler** boiler(Q_flow_max=1300, volume(T(start=328.15, fixed=true)))
- Equations for Object Connection: this part defines the topological connections between physically connected objects, such as a connection between boiler and pump shown in Figure 7: **connect**(pump.port_b, boiler.port_a)

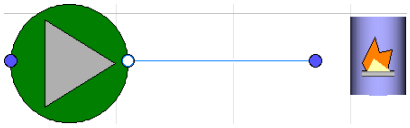

Figure 7: Connection between boiler and pump

Based on the code structure, Modelica code generator will automatically translate SimModel objects into Modelica elements of AixLib according to the mapping rules described in previous section.

EXPERIMENT

Experiment Use Case Definition

We defined a generic SimModel use case based on German guideline VDI 6020-1 that focuses on a heating system of a gas boiler for generation and a radiator located in a single thermal zone (Figure 5). The outcome Modelica .mo file generated for the use case is verified by manually comparing with the baseline model defined in AixLib.

XML Binding Result of SimModel

In our SimModel skeleton:

1. 5 C++ sub-namespaces are defined for storing functional classes of Sim Resource schema, Sim Geometry schema, Sim Core schema, Sim MEP schema and Sim Building schema.
2. 2611 C++ functional classes of SimModel are generated and distributed in above 5 namespaces.

Development Result of System GUI

Figure 8 shows a GUI layout developed for our model tranformation system.

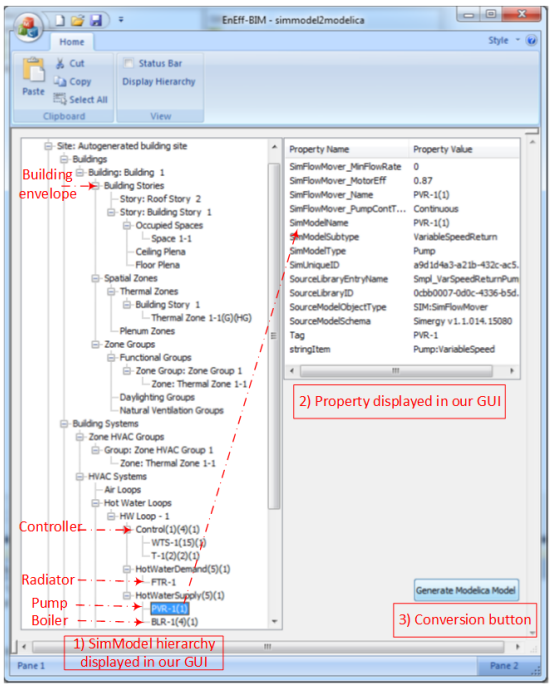

Figure 8: System GUI developed for model transformation from SimModel to Modelica

As shown in the GUI layout: 1) a hierarchy of SimModel components are firstly extracted from SimXML file through calling API functions of SimModel C++ skeleton class. 2) The extracted SimModel hierarchy is then visualized in the GUI as a tree. 3) Property values of specified SimModel components, e.g., the pump selected, will be displayed in the right side, and can be customized by the user. 4) Modelica .mo file will be automatically generated by clicking the lower-right button Generate Modelica Model.

Transformation Result of Modelica Model

In Figure 9, a Modelica code example representing a connection between pump and boiler is generated from the SimConnection entity of SimXML file. As shown in the SimXML code, the direction of hot water flow is determined by connecting the SourcePort to TargetPort. Nonetheless, the corresponding Modelica function connect is generated to support bi-directional water flow.

```
<SimConnection_HotWaterFlow_Default RefId="ID29768">
  <simres:SourcePort>ID31720</simres:SourcePort>
  <simres:TargetPort>ID31725</simres:TargetPort>
  ......
</SimConnection_HotWaterFlow_Default>

connect (pump.port_b, boiler.port_a)
```

Figure 9: Code generated for SimConnection

A Modelica-based BEPS model transformed from the SimModel use case is illustrated in Figure 10. NightSignal is a working schedule of the pump defined in SimModel. const is the temperature set to the boiler. baseParameters is a set of basic parameters defined in Modelica. This use case is verified for demonstrating the transformation system proposed.

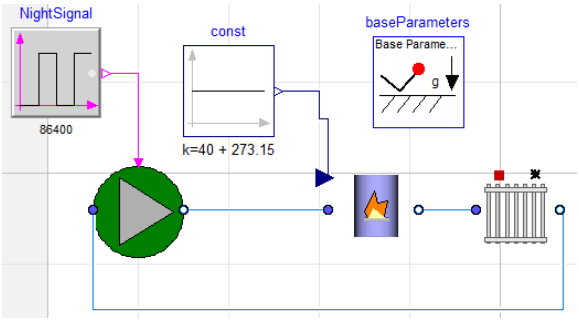

Figure 10: Modelica model generated from the HVAC system defined in SimModel use case.

HVAC component positions, illustrated in the graphical representation (see Figure 10) of .mo file, are determined by their 2D coordinates. At present, we need to assign these 2D coordinates manually for arranging the diagram layout.

CONCLUSION

In this paper, we have developed a new model transformation prototype between BIM and Modelica data models for BEPS purposes. As SimModel is the only BIM format that contains the necessary HVAC data for BEPS and the structure of this data model aligns with the structure of IFC, we use SimModel as a placeholder for IFC in our development. We focus mainly on converting HVAC systems from SimModel into Modelica, in order to accelerate BEPS model development. A generic use case of heating system is adopted to demonstrate this prototype.

The model transformation from SimModel to Modelica presents a unique challenge that must account for significant differences in their respective model structures, such as the model parameter and component definitions, the modeling hierarchy and LOD. The semantic mismatches on model definitions are more difficult, thus, we resolve this by defining parameter and component mapping rules between SimModel and Modelica data models.

As a consequence, the prototype demonstrates the potential to reduce development time of Modelica models by reusing building information data stored in a BIM. Simulation specialists will be more flexible and productive to focus on their creative work such as improving the energy efficiency of a given design rather than to waste time on manually importing data.

FUTURE WORK

This prototype development, mainly driven by the hot water loop use case, will be a first step to make Modelica more accessible to BEPS. Future work needs to stimulate additional development for converting geometric representation of building envelope of Sim-

Model into Modelica objects based on the AixLib. In order to show the generality of the model transformation prototype, additional data type mapping rules will be developed to enhance the transformation of supplementary HVAC components for more complicated HVAC use cases. In the end, the diagram layout of air and water loops will be arranged by automatically assigning 2D coordinates of HVAC components.

ACKNOWLEDGEMENT

In this paper, the work conducted within the German research project EnEff-BIM, Energy Efficient Modeling and Simulation Based on Building Information Modeling, participating in the IEA EBC Annex 60[1], was funded by the BMWi[2] under Contract No. 03ET1177A.
Other parts (no duplications) of the preliminary research work in Ireland were supported by a Marie Curie FP7 Integration Grant within the 7th European Union Framework Programme.

REFERENCES

Aksamija, A. 2012. BIM-Based building performance analysis: evaluation and simulation of design decisions. *Proceedings of the 2012 ACEEE Summer Study on Energy Efficiency in Buildings*, pages 12–1.

Basarkar, M., O'Donnell, J., Haves, P., Settlemyre, K., and Maile, T. 2012. Mapping HVAC systems for simulation in EnergyPlus. In *SimBuild 2012 IBPSA Conference*, Madison, WI, USA.

Bazjanac, V. and Crawley, D. B. 1997. The implementation of Industry Foundation Classes in simulation tools for the building industry. In *Building '97 Simulation Conference*, Prague, Czech Republic.

Bazjanac, V. and Maile, T. 2004. IFC HVAC interface to EnergyPlus-a case of expanded interoperability for energy simulation. In *SimBuild 2004, IBPSA-USA National Conference*.

CodeSynthesis 2011. XSD: XML data binding for C++ [Computer Software]. In *Retrieved from* `http://www.codesynthesis.com/products/xsd/`.

Constantin, A., Streblow, R., and Dirk, M. 2014. The Modelica *HouseModels* library: presentation and evaluation of a room model with the ASHRAE standard 140. In *Proceedings of the 10th International Modelica Conference*, Lund, Sweden.

EBC 2014. Aixlib [Modelica BEPS library]. In *Retrieved from* `https://github.com/RWTH-EBC/AixLib`, Institute for Energy Efficient Buildings and Indoor Climate EBC, Energy Research Center E.ON, RWTH Aachen University, Germany.

Jeong, W., Kim, J., Clayton, M. J., Haberl, J. S., and Yan, W. 2014. A framework to integrate object-oriented physical modelling with building information modelling for building thermal simulation. *Journal of Building Performance Simulation*.

LBNL 2014. Simergy [Computer Software]. In *Retrieved from* `https://simergy.lbl.gov/`, Lawrence Berkeley National Laboratory, USA.

Liebich, T. 2009. IFC 2x edition 3 model implementation guide. In *Retrieved from* `http://www.buildingsmart-tech.org/downloads/accompanying-documents/guidelines/IFC2x%20Model%20Implementation%20Guide%20V2-0b.pdf`.

O'Donnell, J. 2011. Simmodel: A domain data model for whole building energy simulation. *SimBuild 2011, Sydney, Australia, 11/14/2011-11/16/2011*.

Wetter, M. 2009. Modelica-based modelling and simulation to support research and development in building energy and control systems. *Journal of Building Performance Simulation*, 2(2):143–161.

Wetter, M., Zuo, W., and Nouidui, T. S. 2011. Recent developments of the Modelica Buildings library for building energy and control systems. In *Proceedings of the 8th International Modelica Conference*, Dresden, Germany.

Wetter, M., Zuo, W., Nouidui, T. S., and Pang, X. 2014. Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4):253–270.

Wimmer, R., Maile, T., O'Donnell, J., Cao, J., and van Treeck, C. 2014. Data-requirements specification to support BIM-based HVAC-definitions in Modelica. In *BauSIM 2014 Conference*, Aachen, Germany.

Yan, W., Clayton, M., Haberl, J., Jeong, W., Kim, J. B., Kota, S., Alcocer, J. L. B., and Dixit, M. 2013. Interfacing BIM with building thermal and daylighting modeling. In *Building Simulation Conference*.

[1] IEA EBC Annex 60, http://www.iea-annex60.org/

[2] the German Federal Ministry for Economic Affairs and Energy, http://www.bmwi.de/