AN OPEN IFC TO MODELICA WORKFLOW FOR ENERGY PERFORMANCE ANALYSIS USING THE INTEGRATED DISTRICT ENERGY ASSESSMENT BY SIMULATION (IDEAS) LIBRARY

Ando Andriamamonjy[1], Ralf Klein[1], Dirk Saelens[1,2]
[1]Department of Civil Engineering, KU Leuven, 3001 Leuven, Belgium
[2]EnergyVille, 3600 Genk, Belgium

ABSTRACT

This paper presents the development of the open PYTHON IFC interface for the Integrated District Energy Assessment by Simulation (IDEAS) Modelica library. It adapts IFC to be compatible with IDEAS and subsequently generates automatically a graphically represented Modelica model. The interface was tested on a real world facility where the reliability of the data mining process was emphasized. Nonetheless, it was pointed out that an improved IFC schema towards Building Energy Simulation can improve the presented interface.

INTRODUCTION

Considering the global energy use concern in buildings, more elaborate tools for building energy simulations are designed.
In this context, (Baetens et al. 2015) developed the Integrated District Energy Assessment by Simulation (IDEAS) library. IDEAS is a Modelica (Elmqvist 1978) library which allows prototyping for design and operation of district energy and control systems. It is an open library relying on a physical modelling approach and integrates multi-zone thermal building energy simulations, including both building envelope and heating, ventilation and air-Conditioning (HVAC) systems, and electric system simulations (Baetens et al. 2015).
In the case of a real neighbourhood modelling, IDEAS is used to model each individual buildings and requires a manual input of parameters into the components. A time consuming and error prone task.
A solution is to draw the data from the Building Information Model (BIM) and populate in an automatized or semi-automatized way the IDEAS library components.
The use of BIM, in this context, is motivated by its wide utilization across Europe where it is even adopted as regulation in some European countries. Furthermore, the open-BIM format Industry Foundation Classes (IFC) provides an open and BIM-software independent platform suitable for an open-BIM interface for IDEAS.
A prevalent research in the development of open framework using IFC for BES modelling in Modelica is the work of (Remmen et al. 2015; Wimmer et al. 2015). They present a framework capable of generating automatically a Building Energy Simulation (BES) model including the building shell and systems and it is intended to be compatible with the Modelica libraries developed under the IEA EBC Annex 60 (Energy in Buildings and Communities (EBC) 2014). Nonetheless, the model is generated without a graphical representation.
The current paper presents the development of an IFC interface dedicated to the IDEAS library to semi-automatized the implementation of a BES model. It adapts the IFC data into a format compatible with the IDEAS library components and subsequently generates automatically a BES Modelica model. The main features of the interface are the generation of a graphically represented BES model, and its compatibility with the current version IFC2X3 as well as with the new version IFC2X4 of IFC.
As the IDEAS library development is an ongoing project, the interface is developed synchronically with the IDEAS library development to avoid incompatibilities.
This paper consists of two major parts: The approach and methodology section defines the type of data required by IDEAS components and presents the IFC data query process to suit the requirements of the IDEAS components. A detailed explanation of the BES model auto-generation is also presented in this section. In the illustrative application section, the interface is applied to a real world test facility having an as-built BIM model where the practical use of the interface is shown. In addition, the generated Model is compared with measurement data to assess the data retrieval process reliability. A discussion on the future works and limitations of the interface followed by a conclusion ends this paper.

APPROACH AND METHODOLOGY

Modelica is a component oriented programming language where a model consists of an interconnected components. The IDEAS library contains top level components which can be used to model individual building response as well as an entire district.

As the present interface adapts IFC data into IDEAS component parameters, it is assumed that the reader is familiar with the IDEAS library (especially with the *IDEAS.Buildings* package) and with the IFC2X3 schema. A detailed information on IDEAS can be found on https://github.com/open-ideas while IFC2X3 schema is explained in (Hafele et al. 2010).

The current approach focuses on the building level and mainly on the building envelop modelling with a limited HVAC system. For this purpose, the *IDEAS.Buildings.Components* package contains validated components representing the main features of a building envelope: the zones (*.Zone*), the outer wall (*.OuterWall*), the internal wall (*.InternalWall*), the slab on ground (*.SlabOnGround*), the window (.Window) and the boundary wall (*.BoundaryWall*). The *IDEAS.HeatingSystems* package contains heating systems components.

To build an IDEAS based BES Model, it is necessary to know the building layout information which provides the components number and type that need to be declared and their relationship to each other. In a manual modelling process, the building layout information is obtained by analysing visually the buildings drawing or its virtual representation. For instance, considering the virtual representation of a simplified building presented in the figure 1, its detailed IDEAS model consists of declaring two zones, six external walls and one internal wall component. The Modelica *connection* is then implemented by considering the boundary type (e.g. Internal wall, external wall,…). Subsequently, the parameters describing the building elements such as the geometrical characteristics, physical and thermal properties (construction type, materials, windows type, …) are inserted into the components.

Besides, a common way to implement a top level Modelica model is to *drag and drop* the graphically represented components into the graphical interface of the Modelica simulation engine, insert the parameters and connect the component graphically. This top level model is a representation of a set of text lines following the Modelica language syntax (Modelica Association 2012) which is converted into a graphical representation by a Modelica simulation engine such as OpenModelica (by PELab, Linköping University) or Dymola (by Dassault Systèmes).

The IFC into IDEAS semi-automated approach consists then on:

(1) Retrieving the building layout information, the geometrical information and the physical properties from IFC (namely IFC2X3).

(2) Generating a set of text in accordance with the Modelica specifications (Modelica Association 2012) and the IDEAS library structure where all the data previously retrieved are inserted.

Throughout this paper, a parameter related to an IDEAS component is noted as *.component.parameter* (e.g. *.zone.nsurf* represents the *nsurf* parameter of a zone component). To represent IFC objects, σ_i defines an *IfcSpace*. ε_{ij} represents the j^{th} boundary (*IfcRelsSpaceBoudanry* object) of the space σ_i and ρ_{ij} (*IfcRel sSpaceBoudanry.ConnectionGeometry*) its geometrical representation (coordinate and Area). E_t (*e.g. IfcWallStandardCase, IfcWindow,…*) represents the architectural object and H (only *IfcSpaceHeaterType* is investigated) a HVAC system object. To refer to an IFC object attribute (explicit or inverse), the following annotation is adopted: *IfcObject.attribute*.

Figure 1 relates a simple building with two spaces σ_1 and σ_2 which will serve as a demonstration case throughout this section and referred as BIM_1.

Building Layout and geometrical data. The IFC features space definition and space boundary are used to generate the building layout information. A *space* is an enclosed volume physically bounded by architectural elements (walls, windows, roof, ceilings,...) with a vertical extent from the floor to the ceiling. Its definition assigns an *IfcSpace* object to each zone defined in the BIM.

A *Space boundary* defines the relationships between the space (*IfcSpace*) and the IFC objects representing the building elements (*IfcWall, IfcWindow,…*) that enclose it.

The definition of spaces is required prior to the space boundary definition. The space boundaries are then used as a baseline information to build the building layout information.

Space boundary is implemented on the IFC model by the use of the *space boundary add-on view* Model View Definition (MVD, see (Eastman et al. 2011) for further information on MVD) (Weise et al. 2009) or alternatively, by using the Space Boundary Tool (SBT) (Rose & Bazjanac 2015) from an IFC file containing only space definitions.

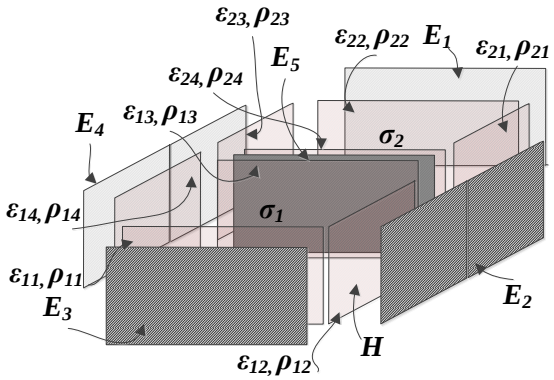

Figure 1 Two zones simplified building

Starting from the space and space boundary definition in the IFC model, a set of rules has been developed to map IFC objects into IDEAS components and can be summarized as follow: (subscript a and b are used for generality while the numbers refer to figure 1)

1) An IFC object is categorized as a *.Zone* (*IDEAS.Buildings.Components.Zone*) component if it

is an *IfcSpace* object (e.g. σ_1 and σ_2). Its identifier is retrieved from the *IfcSpace.Name* attribute. The number of boundary related to a zone is obtained by identifying the number of *IfcRelSpaceBoundary* (ε_{ij}) object related to the *IfcSpace.BoundedBy* attribute.
The number of zone in the model, the name of each zone and the number of boundary (*.Zone.nsurf*) related to each zone are identified through this step. The zone volume (*.Zone.V*) is defined as well by retrieving the *IfcSpace.IsDefinedBy.RelatingPropertyDefinition.GrossVolume* attribute of the space object.

2) An IFC object E_t is considered as an *.InternalWall* component (*IDEAS.Buildings.Components.InternalWall*) if : (a) E_t is a common boundary between two adjoining spaces σ_a and σ_b which means in IFC schema that E_t is a common *IfcRelSpaceBoundary.relatedbuildingelement* attribute between a boundary ε_{aj} of σ_a and a boundary $\varepsilon_{bj.}$ of $\sigma_{b.}$ (b) The intersection area ($\rho_{aj} \cap \rho_{bj}$) between the boundary ε_{aj} and ε_{bj} is not null.
Considering the figure 1, E_5 is related to ε_{13} and ε_{24} and $\rho_{13} \cap \rho_{24}$ is not null. Therefore, E_5 is an internal component between σ_1 and $\sigma_{2.}$
Through this step, the object Name is retrieved to be used as the component identifier (i.e. name of the *.InternalWall* component in the generated model). The intersection area is computed for the wall area parameter (*.InternalWall.Awall)*. The construction type *(.InternalWall.constructiontype)* is given by the name of the *IfcMaterialLayerSet.LayerSetName* object related to E_t (an explanation on the construction type package implementation is given further). A data structure ({{ E_t , σ_a },{E_t, σ_b }}) is generated to represent the connection between E_t, σ_a and $\sigma_{b.}$
Default values are used for the remaining *.InternalWall* parameters, especially for the wall inclination (*.InternalWall.inc =90°*) as only vertical wall is supported.
Internal opening such as door object (*IfcDoor*) is modelled as an internal wall as long as it agrees with the rules 2.a and 2.b.

3) E_t is mapped as an *.OuterWall* component if it is a subtype of an *IfcWall* or an *IfcRoof* object related to a space boundary ε_{ai} of a space σ_a and does not meet the conditions of (2). For example, E_2 is related to ε_{12} and ε_{21} but ρ_{12} and ρ_{21} do not intersect. Thus, E_2 connects σ_1 to the outside and the common area is ρ_{12} .
As for previously, the object name is used as component identifier. The external wall area (*.OuterWall.Awall)* is computed from ε_{ai} and the construction type retrieved from *IfcMaterialLayerSet.LayerSetName.* A data structure {{ E_t, σ_a },{E_t, Ext}} is generated to represent the connection of E_t to σ_a and to the outside condition.
If E_t is a subtype of the *IfcWall* object, the angle between the south and the vector normal to E_t is computed for the wall azimuth (*.OuterWall.azi)* where only vertical wall is assumed (*.OuterWall.inc =90*) .If E_t is an *IfcRoof*, the parameter *.OuterWall.inc =0* and the default value is used for *.OuterWall.azi.*
In this process, external doors (*IfcDoor* object) are modelled as *.OuterWall* components where the door characteristics are assigned to its construction type parameter.

4) An object E_t is a *.SlabOnGround* if: (a) it is a slab object (*IfcSlab*) related to a boundary ε_{ai} of a space σ_a located in the lowest floor of the building. (b) It is geometrically located at the bottom of $\sigma_{a.}$The *.SlabOnGround.PWall* is computed based on the perimeter of the zone $\sigma_{a.}$

5) E_t is a *.Window* if it is an *IfcWindow* object and do not meet the conditions of (2). Window glazing type (*.Window.glazing*) is assigned according to the window name (explanation on glazing package generation is given later) while default values are used for the frame type (*.Window.fraType*), shading type (*.Window.shaType*) and frame fraction (*.Window.frac*).

6) E_t is an *IDEAS.HeatingSystems* component if it is an HVAC object (Currently only *IfcSpaceHeater* supported) and geometrically located in a space σ.

Considering these rules, the building layout of the BIM_1 is schematized in figure 2. The lines (arrows) represent the connection between the components and will be used as a pattern to implement the Modelica connection.

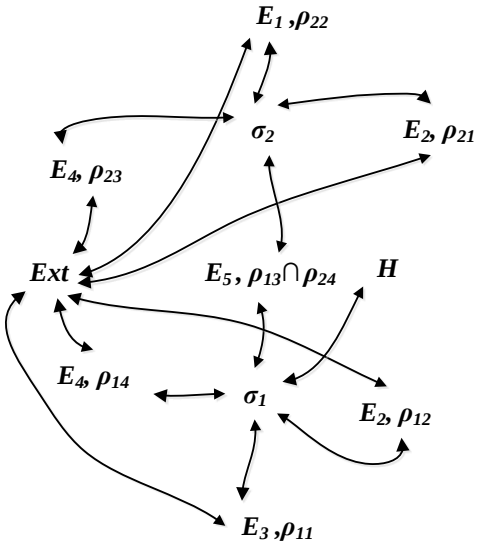

Figure 2 Building layout of BIM_1

The building layout information is the cornerstone of this approach and a validation process was undertaken to ensure its accuracy. The Solibri Model Viewer (by Solibri, a Nemetschek company) was used for a visual comparison between the original BIM and the generated building layout graph (e.g. Figure 2). For each space and building objects a rigorous visual check was performed.
The validation started with a relatively simple building. The complexity of the building's

architecture was increased step by step by introducing new features (new floor, windows, oriented walls,…) to test if all these features are correctly translated.
Furthermore, to ensure the code robustness, it has been tested on real world BIM model such as the BIM models developed by (Karlsruhe Institute of Technology 2013).

Physical Properties. Construction type is assigned to an *IDEAS.Buildings.Components* (e.g. *.OuterWall* or *.InternalWall*) by declaring the *.ConstructionType* parameter from the *IDEAS.Buildings.Data.Constructions* package (see https://github.com/open-ideas). A construction type is defined by the number, the material name and the thickness of the constituting layers present in the construction. The definition of construction type from IFC starts from the *IfcMaterialLayerSet* object related to the object E_t. It represents a type of construction and contains the construction name through the *IfcMaterialLayerSet.LayerSetName* attribute while the *IfcMaterialLayerSet.ForelayerSet* attribute contains a set of *IfcMaterialLayer* object. The number of layer in the construction is obtained by identifying the number of *IfcMaterialLayer* in the set. Moreover, an *IfcMaterialLayer* object represents a material layer and provides the material name (*IfcMaterialLayer.Material*) and the layer thickness (*IfcMaterialLayer.LayerThickness*). A construction type is therefore added to the *IDEAS.Buildings.Data.Constructions* package based on the aforementioned information.
The materials in each construction type is defined from the *IDEAS.Buildings.Data.Materials* package. Each material is described according to its thermal conductivity, thermal capacity, density, and emissivity (long wave and shortwave). As these information are not available in the IFC2X3 schema, they need to be integrated manually. Nonetheless, the manual input of these data is facilitated by inventorying the *IfcMaterial* instance in the IFC Model. *IfcMaterial* object represents a material type and allows to know the number and the names of material in the BIM or IFC model. It can be used to generate a file listing the material names. Users can write the properties on the file and use it as an input to a code routine which adds the material into the *IDEAS.Buildings.Data.Materials* package.
Glazing type for windows are defined in the *IDEAS.Buildings.Data.Glazings* package. A glazing type is defined by the material layers (gas and glass), the glass and gas type, thickness, u value, g value and the optical properties such as the absorbed and transmitted solar radiation. As such information are missing from the IFC2X3 schema as well, a code routine capable of adding a glazing component to the *IDEAS.Buildings.Data.Glazings* package from the window 7 software (Arasteh et al. 1994) (by Lawrence Berkley National Laboratory) output was designed.

Heating and ventilation system. At the current status of the work, HVAC system modelling is limited to the identification of the zones connected to a HVAC system. Generic IDEAS HVAC components were used to represent the HVAC system.

Model Generation. A Modelica model is a set of text complying with the Modelica language specification (Modelica Association 2012). A component declaration is similar to variable declaration in low level programming language such as C/C++ where the component type is followed by the component name and the parameters assigned as arguments. For example, *IDEAS.Buildings.Components.Zone zone1(nSurf=2,V=2)* declares an IDEAS zone component named zone1 connected to two surfaces and having a volume of 2 m^3. Component declaration is followed by a Modelica annotation which defines its graphical representation and position in the graphical interface of the top level Modelica model.
To represent the relationship between the components, the built-in Modelica *connect* function is used. It represents generally a flow between two components (current, fluid, ...) and is graphically represented as a line.
Therefore, to generate an IDEAS Modelica model, a template, representing the semantic of each components in the *IDEAS.Buildings.components* and *IDEAS.HeasingSystem* package, is required. A template is defined as a component instance where the parameters values are replaced by keywords put between delimiters. For example, *IDEAS.Buildings.Components.Window #WindowName# (A=#WinArea#, inc=#WinInclination#, azi=#WinAzimuth#, redeclare parameter IDEAS.Buildings.Data.Glazing.#GlazingType# glazing) annotation (Placement(transformation(extent={{#x_{pos}#,#y_{pos}#},{ #x_w#,#y_h#}})))* defines a basic template for IDEAS window model where the parameters between "#" need to be replaced by the real values retrieved previously from IFC.
The parameters *#x_{pos}# and #y_{pos}#* define the position of the component in the graphical interface while *#x_w# and #y_h#* represent respectively the component width and height.
The building layout information is used to declare the components needed to describe the BES model. For example, considering the figure 2, two zone components, one internal wall component and six external wall components are declared automatically by iterating the template definition through the layout information. The component annotation is modified as well in a way that zones on the lower floor are placed on the lower position (y_{pos} =0) while the first zone of each floor is placed on the left position (x_{pos}=0 and y_{pos} =0 for the first zone of the lower floor). Windows and external walls related to a zone are placed on the left while common walls are graphically

placed between the two zones. Finally, parameters between delimiters are replaced by their real values.
Newly declared components are connected automatically based on the data structure obtained from the building layout (e.g. Figure 2). *IDEAS.buildings* components (IDEAS v.0.2) use one port to define the flow between two components. These connections are implemented by using the keyword connect where the following expression is used for the graphical representation of a connection between two components: *annotation (Line(points={{#x_{comp1}#,#y_{comp1}#},{(#x_{comp1}# - #x_{comp2}#)/2, #y_{comp1}#},{(#x_{comp1}# - #x_{comp2}#)/2, #y_{comp2}#},{ #x_{comp2}#,#y_{comp2}#}},color={255,204,51}, thickness=0.5, smooth=Smooth.None))*; *#x_{compi}# and #y_{compi}#* represent the coordinate of the *port* of the component i in the graphical interface and it is used to represent graphically the connection between two components.

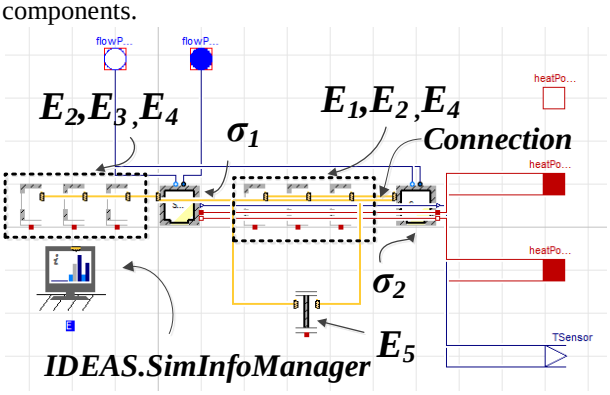

Figure 3 Auto generated BIM_1 BES model

In addition, the *IDEAS.SimInfoManager* component which loads the weather data files and computes the solar irradiation on the zone surface is declared. It is initialized with the default parameters (TMY3 file location, latitude, longitude, …) where the users have to insert these information subsequently. Furthermore, the ports (*heatPortCon, heatPortRad, heatPortEmb, flowPort_In and flowPort_Out*) which allows to connect the BES model to an HVAC are generated as well.
Figure 3 presents the auto generated IDEAS model from the BIM_1 where σ_i represent the *.Zone* components, $E_{i,\ i\ \epsilon\ [1,4]}$ the *.OuterWall* components and E_5 an *.InternalWall* component.
To test the interface robustness, it has been tested on various real world BIM such as the duplex apartment BIM model which can be found on (BuildingSMARTalliance 2012).

On the generated model. Default values are assigned to parameters not given throughout this process, thus the generated model can be compiled and simulated directly from any Modelica simulation engine compatible with the IDEAS library. Therefore, it constitutes a starting point for the implementation of a more elaborated BES model with complex HVAC system and control.
The process generates a BES model based on a zone to zone mapping. Although with a high level of detail, it may increase the computation time for a large district simulation. A next step of this work is to aggregate the *IfcSpace* objects having the same characteristics into an unique zone component.

On the interface. Although the new IFC standard (IFC2X4) was issued, the interface relies on the IFC2X3 format as it is currently the widely used and supported format among the BIM authoring tools (BuildingSMART-certification 2016). Nonetheless, a migration towards IFC2X4 is envisioned with an upgrade of the data extraction process as IFC2X4 provides more features for BES modelling.
Besides, the interface was implemented with the open programming language PYTHON (Van Rossum & Drake 2001) while the IfcOpenShell (IfcOpenShell 2015) library was used to handle the IFC file.
The interface is currently designed for the IDEAS library but can be adapted to other BES dedicated Modelica library such as the Modelica Buildings library developed by the LBLN (Wetter et al. 2014).

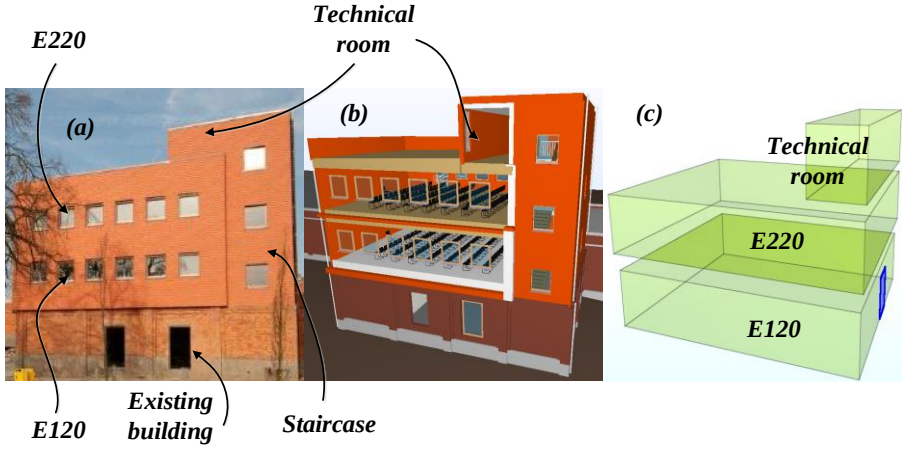

Figure 4 (a) test facility, (b) as-built BIM model, (c) space definition

ILLUSTRATIVE APPLICATION

The interface has been applied on a real world test facility where its practical use is presented in this section.

As the generated model accuracy relies on the translation of IFC data into IDEAS input, it is interesting to assess the accuracy of the directly generated model towards the reality under a specific known condition where the building layout, the properties and the geometrical data will constitute the influential factor on the building thermal response. Therefore, a comparison of the simulated data and measured data during a known experiment condition is also presented in this section.

The test facility. The test facility (figure 4.a) is located in Ghent (Belgium) on the KU Leuven university campus. It is constructed on top of an existing university building and has four rooms which are the two lecture rooms (E120 and E220 on the first and second floor), the staircase and the technical room. The lecture rooms form two insulated boxes. Room E120 has massive brick walls and room E220 consists of a light weight timber frame structure. The indoor parameters such as temperature, humidity and CO_2-level within the lecture rooms are monitored. The facility is also equipped with its own weather station measuring the global solar irradiation, relative humidity, temperature, precipitation, wind speed and wind direction (see (Andriamamonjy & Klein 2015) for more information on the facility characteristics). Furthermore, an as-built BIM (figure 4.b) model is available for building operation purposes and will serve as an input for the interface.

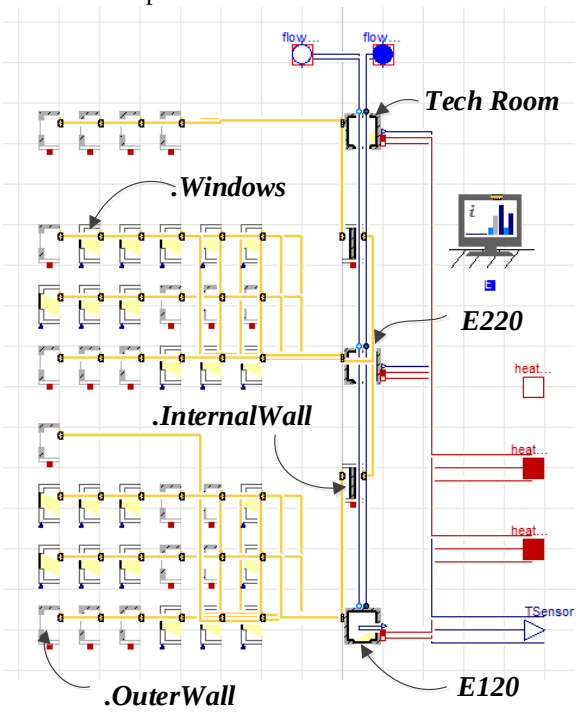

Figure 5 Auto-generated BES model of the facility

Practical implementation. The first step is to define the spaces in the as-built BIM. As the two lecture rooms are assumed to be thermally insulated from the staircase and the ground floor; spaces were only defined in the lecture rooms and the technical room (Figure 4.c).

To obtain an IFC format with space boundary definition, the BIM was exported into an IFC2X3 file from the BIM authoring tool (namely Revit 2016 by Autodesk) by using the Coordination View 2.0 MVD (Hafele et al. 2010) with the space boundary add-on view. Alternatively, the extended Facility Management Handover (FM Handover) MVD could be also used (buildingSMART 2009).

The exported IFC file is then utilized as an input to generate a building envelope model from the PYTHON code routine based on the mapping rules developed earlier.

A .csv (comma separated values) format file containing the list of the materials used is generated automatically as well. It is used to insert the material properties from the delivery manual and to generate a material type in the *IDEAS.Buildings.Data.Materials* package.

The window glazing type was remodelled in window 7 (Arasteh et al. 1994) based on the delivery manual data and used as input to create the corresponding glazing type in the *IDEAS.Buildings.Data.Glazings* package. Figure 5 shows the final generated model. One can note that each object (e.g. *IfcWindow*, *IfcWall*) present on the BIM is represented as an individual component in the Modelica model.

Model assessment. A controlled experiment was performed on the facility during three days (26 to 29 December 2015). The rooms air temperature were monitored while the heating/cooling were off. The access was restricted while doors were temporary sealed and windows closed. The goal was to assess the building thermal response towards the outside condition.

The facility is an experimental building where the material and glazing properties are well known and assumed to be correct. Thus, an important discrepancy between the measurements and the simulation can only indicates the unreliability of the IFC data into IDEAS conversion.

Figure 6 shows the comparison between the simulated values (from the model on figure 5) and the measured values from the two lecture rooms. One can note the similarities in the curves dynamics with an average discrepancy of 0.3 °C during the three days. This discrepancy might be due to the default values assigned to some parameters (e.g. Window frame type, and frame fraction). In addition sensors inaccuracy need to be taken into account as well. Nonetheless, it could be concluded that the auto-generated model depicts the reality and the data

retrieval process from IFC into IDEAS can be considered as reliable.

DISCUSSION

On the advantages. The workload reduction due to the interface use cannot be quantified precisely. Nonetheless, its utilization avoid the manual and time consuming process of data collection, dragging, dropping components and implementing the connections. Furthermore as the generated model has a graphical representation, it provides a certain flexibility to the user for modification and then adding the model to an existing district model.

On the future improvements. Although the buildings are modelled with a high level of details (e.g. see figure 5), the contribution of such details to the model accuracy could be negligible but increases the computation time. A next step is to aggregate all the external windows or construction having the same type (glazing or construction type) and orientation into an unique component.
Besides, the current work focused on the building envelope. A next step is to improve the data retrieval process for HVAC modelling in IDEAS.

exported as a generic *IfcBuildingElementProxy* object. Hence, a standard, which is able to remove the limitations of IFC2X3 and is dedicated to BES is required.
In addition, the full adoption of an IFC4 MVD by BIM authoring software will also require some time. The certification procedure that has to guarantee the quality of the exported data will take additional time.
Considering these issues, a robust strategy for a BIM to Modelica BES modelling has to use the currently supported and certified standard MVDs. An easy adaptation of the workflow has to be possible seen the ongoing BIM technology evolution.

CONCLUSIONS

This paper presents an open IFC interface for the IDEAS library implemented in PYTHON and capable of generating semi-automatically a BES model. It is an automatized data mining and model implementation process which retrieves the building layout, the geometric data and the physical properties from an IFC2X3 file. It adapts these data into IDEAS components compatible format and generates automatically a graphically represented Modelica model. The interface was applied on a real use test facility where the method reliability was emphasized.

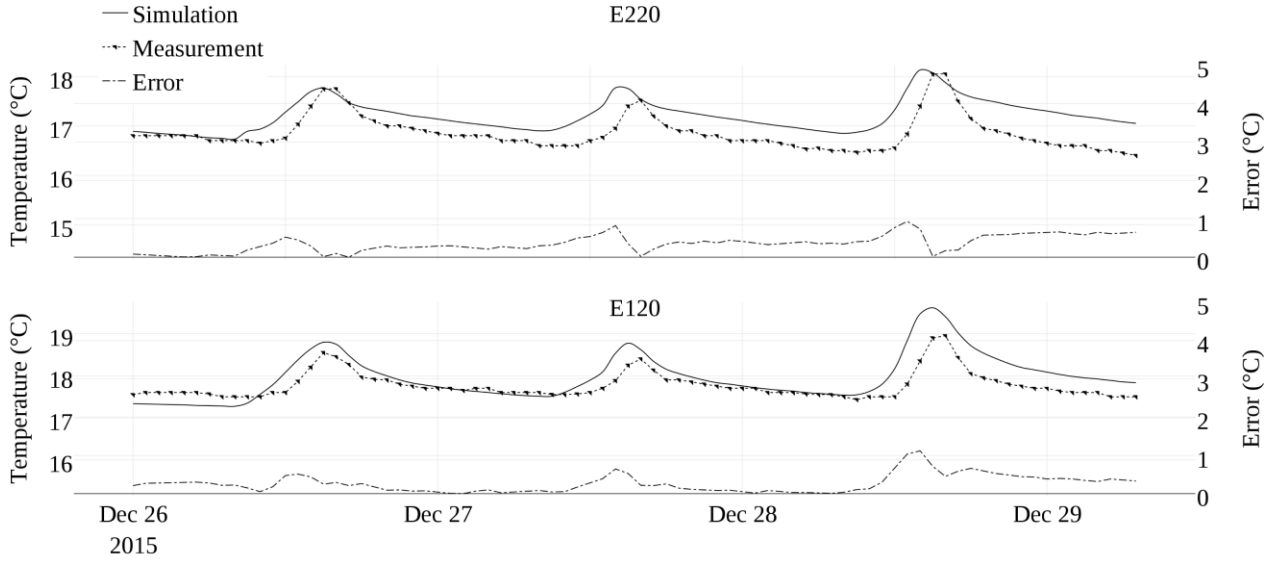

Figure 6 Simulation and measurement comparison

On the IFC format. Due to its limitations, the IFC2X3 cannot be used for a fully automated implementation of IFC into IDEAS. It emphasizes the need of an improved MVD dedicated for BES. For instance, if the material properties were integrated within the IFC model, the manual input of material and glazing properties could have been skipped and reduce even more the BES modeller workload. In addition, a lack of semantic for HVAC has been observed, for instance, no specific IFC object has been defined for Variable Air Volume Box system which is

Nonetheless, it has been pointed out that a simplification of the auto generated model structure is required to reduce simulation time while the enrichment of the IFC format can improve the automation process and achieve a fully automatized model implementation.

ACKNOWLEDGEMENT

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) Program. 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

Andriamamonjy, A. & Klein, R., 2015. A modular, open system for testing ventilation and cooling strategies in extremely low energy lecture rooms. In *36th AIVC- 5th TightVent- 3rd venticool*.

Arasteh, D., Finlayson, E. & Huizenga, C., 1994. Window 4.1: A PC Program for Analyzing Window Thermal Performance in Accordance with Standard NFRC Procedures. Available at: https://windows.lbl.gov/software/window/window.html.

Baetens, R. et al., 2015. OpenIDEAS - An Open Framework for Integrated District Energy Assessment. *Proceedings of the 14th IBPSA Conference - Building Simulation 2015*, (November).

buildingSMART, 2009. *FM Handover View Aquarium*, Available at: file:///C:/Users/u0097128/Downloads/20090805_FM_Aquarium_Final.pdf.

BuildingSMARTalliance, 2012. Common Building Information Model Files and Tools. Available at: https://www.nibs.org/default.asp?page=bsa_commonbimfiles.

buildingSMART-certification, 2016. IFC certification 2.0. Available at: http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc2x3-cv-v2.0-certification/participants [Accessed January 1, 2016].

Eastman, C. et al., 2011. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors* 2nd ed., John Wiley & Sons, Inc., Hoboken, New Jersey.

Elmqvist, H., 1978. *A structured model language for large continuous systems*. Lund Institute of Technology. PhD Thesis.

Energy in Buildings and Communities (EBC), 2014. IEA EBC Annex 60. Available at: http://www.iea-annex60.org/.

Hafele, K.-H., Geiger, A. & Liebich, T., 2010. Coordination View Version 2 . 0 for IFC 2x3. Available at: http://www.buildingsmart-tech.org/downloads/view-definitions/coordination-view/sub-schema/CoordinationView_V20_EntityList_IFC2x3_Version16_Final.pdf.

IfcOpenShell, 2015. IfcOpenShell. Available at: http://ifcopenshell.org/.

Karlsruhe Institute of Technology, 2013. Karlsruhe Institute of Technology: Semantic Data Models. Available at: http://www.iai.fzk.de/www-extern/index.php?id=1135&L=1.

Modelica Association, 2012. Modelica ® - A Unified Object-Oriented Language for Systems Modeling Language Specification. Available at: https://www.modelica.org/documents/ModelicaSpec33.pdf.

Remmen, P. et al., 2015. An open framework for integrated BIM-based building performance simulation using Modelica. *Proceedings of the 14th IBPSA Conference - Building Simulation 2015*.

Rose, C.M. & Bazjanac, V., 2015. An algorithm to generate space boundaries for building energy simulation. *Engineering with Computers*, 31, pp.271–280.

Van Rossum, G. & Drake, F., 2001. Python Reference Manual. Available at: https://www.python.org/.

Weise, M. et al., 2009. Space Boundaries for thermal analysis. , (August), p.68. Available at: http://www.buildingsmart-tech.org/downloads/accompanying-documents/agreements/IFC2x3-space-boundary-implAgreement-2009-09-17.pdf.

Wetter, M. et al., 2014. Modelica Buildings library. *Journal of Building Performance Simulation*, 7(4), pp.253–270.

Wimmer, R. et al., 2015. Implementaation of advanced BIM-based mapping rules for automated converions to Modelica. *14th Conference of International Building Performance Simulation Association, Hyderabad, India*.