CIB World Building Congress 2019
Hong Kong SAR, China
17 – 21 June 2019

SIMULTAN as a Big-Open-Real-BIM Data Model – Evolution of Virtual Building from Design through Construction into Operation Phase

Galina Paskaleva,
Research Unit of Building Physics,
Institute of Material Technology, Building Physics, and Building Ecology, TU Wien, Vienna
(email: galina.paskaleva@tuwien.ac.at)
Thomas Lewis,
Research Unit of Building Physics,
Institute of Material Technology, Building Physics, and Building Ecology, TU Wien, Vienna
(email: thomas.lewis@tuwien.ac.at)
Sabine Wolny,
Research Unit of Building Physics,
Institute of Material Technology, Building Physics, and Building Ecology, TU Wien, Vienna
(email: sabine.wolny@tuwien.ac.at)
Thomas Bednar,
Research Unit of Building Physics,
Institute of Material Technology, Building Physics, and Building Ecology, TU Wien, Vienna
(email: thomas.bednar@tuwien.ac.at)

Abstract

The goal of Building Information Modeling (BIM) is the uninterrupted use of digital construction models from the planning stage onwards. As the operation phase approaches, the involved processes become increasingly complex. Stakeholders work at varying pace and utilize highly specialized tool chains that need to communicate seamlessly with each other. BIM aims to satisfy those needs. However, even the most extensive Open BIM standard – IFC- does not provide the necessary data structure, in spite of its more than one thousand different types, because its focus is the planning phase. Our approach, SIMULTAN, in contrast to IFC, consists of less than 50 different basic types. They can be dynamically combined to increasingly complex models, which can themselves be used as prototypes for other models. This enables each domain expert to create custom data structures for any specific task, which are automatically compatible with data structures of other stakeholders. In this way, unlike IFC, our data model removes the long wait for the next implementation cycle to be completed by developers. Furthermore, it shortens the training time and facilitates the loss-, corruption-, and conflict-free exchange of information between domain experts, which is a key aspect of BIM.
As proof of concept, we present the evolution of the virtual building model from the construction to the operation phase. In each phase, stakeholders enrich the data model with the information they need for, e.g., analyses. Our approach considers the needs of the operation phase and works towards the construction and commissioning phases to set up the necessary data infrastructure by collecting information at every stage, from every stakeholder and considering inter – dependencies, potential conflicts and threats to model consistency.
Results show that the flexible data model of SIMULTAN can be adapted to the task. It can handle structured as well as unstructured data.

Keywords: data model, construction phase, operation phase, BIM, MSR

1. Introduction

Digital models of buildings are not only important during the design and planning phase. These models should be available to use during construction, operation and onwards, as a dynamic view of processes, in order to maintain an up-to-date overview throughout the lifecycle of the building (Borrmann et al, 2015). The aim of Building Information Modeling (BIM) is to support the use of such digital building models (Borrmann et al, 2015).

An existing well-known standard for BIM is the Industry Foundation Classes (IFC) (buildingSMART, 2018) definition. The IFC models hold geometric data as well as metadata about building objects and it aims to support interoperability. Nevertheless, its main focus is on support for the design phase and of simple data exchange for the duration of the building's life cycle. Goedert et al. (2008) extend the use of BIM throughout the construction phase by additionally accommodating construction documents. Another approach that also tackles the lack of continuity along the building's lifecycle is described in (O'Sullivan et al, 2004). The authors create a framework, incorporating IFC, for monitoring, analyzing and controlling a building throughout its lifecycle based on a set of performance metrics. However, most approaches focus on specific phases of the building's lifecycle and do not consider the entire workflow.

In this paper, we focus on the uninterrupted support for the construction phase and into the operation phase by a new data model, SIMULTAN (Paskaleva, Wolny and Bednar, 2018), which was developed as a big open BIM implementation. We demonstrate that our data model supports the tasks and processes typical of the construction phase and the model can transition seamlessly into support for the operation phase. The support of the design phase was already shown in (Paskaleva et al, 2019).

The remainder of the paper is structured as follows. In the next section, we present typical needs and requirements from the design phase of a building that carry over to the operation phase. In Section 3, we describe the features of our data model that meet those requirements. Section 4 shows a first proof of concept of our approach. Finally, we conclude this paper and give an outlook to our next steps.

2. Typical Tasks from Design to Operation Phase

When is a building well built? Hens (2012) advocates a so-called performance based building design where performance can be defined through a set of parameters that can be calculated in advance. In this context, the performance focuses mostly on energy cost performance. Some sectors have already achieved a high predictability of their planning, allowing them to offer performance guarantees, such as guaranteed minimum yields for photovoltaic or solar plants. Thus, deviations in performance can be quickly attributed to single causes, such as faulty modules, incorrectly placed sensors etc. relying on a digital model of the entire plant, which often also comprises the users.

Such a high reliability and predictability is also the aim for entire buildings or even a complex of several buildings over their complete life cycle. Sustainable planning and use of buildings therefore inevitably involves

1. a digital model of the building
2. intelligent performance monitoring.

In the following brief example, we outline the importance of monitoring over the entire building's life cycle. Let us suppose that the facility manager of a large office building has an estimation of an annual budget for energy cost based on information from the planning phase. The building owner - a large international company, who in this case is also the user of the building, may set the room temperatures individually, by room. During the third year of operation, a new facility manager joins. By studying

the planning figures, this manager realizes that the energy cost results are 25% higher than expected, according to the life cycle analysis at the planning stage. Higher prices are not the reason, but evidently higher demand.
In detail the discrepancy splits into

1. a heating demand, which is 35% higher than planned
2. an electricity demand, which is 20% higher than planned

In a next step, the reasons and the responsible stakeholders are identified. Planners could propose that the reasons are unusual user behavior (e.g., room temperatures set too high), incorrect system operation or incorrect heat measurement. The facility manager could point to other reasons, such as (i) the performance of thermal hull of the building is lower than planned and (ii) components of the ventilation system (air ducts, fire dampers, size of the ventilation plants etc.) were not dimensioned adequately to their actual output.

As a first measure, an energy consultant is contacted to identify the reason. The consultant needs

1. the planning documents of the building
2. confirmation that the building has been built as planned (any commissioning documents)
3. any building simulations that have been performed
4. monitoring data

The data described above could have shortcomings, such as (i) scarce documentation of the simulation, (ii) settings for the plant operation differ highly from those for the simulation, or (iii) the consultant does not use the simulation software the planner has used. Thus, models have to be recreated and work has to be done again.

In order to reduce this effort, that can produce additional mistakes, various points should be considered already at the design phase.

A monitoring concept is usually already part of the planning. It specifies which zones, circuits and networks are monitored by which kind of monitoring units, taking into account possible hierarchies between them. Each unit has a specific location and measuring scope, e.g., for electrical meters that is the list of connected monitored electrical loads.

Each monitoring unit generates data records. A valid data record generated by such a unit comprises

1. a time stamp
2. a data block of measured data (values and units)
3. the id of the monitoring unit (or subunit)

Based on this type of information, monitoring and error prevention should be easier during the operation phase. Features are needed make it possible to determine to what extent the planning corresponds to the actual building performance after the planning and construction phase are completed. Therefore, an ideal system for supporting the operation phase should allow for the digital model features listed below.

Feature 1 Regular performance assessment
Comparison of simulation results versus monitored data should be straightforward. The digital model represents the building in its current state. This means, all deviations from the design at the end of the planning phase are integrated. A building simulation based on monitored input, data such as climate, or any user data (such as presence) etc., can be run at any time. Monitored data is compared against the simulation results. Deviations are automatically displayed to the facility manager, e.g., deviations above a certain threshold are included in the automatic report.

Feature 2 Component based information
It is possible to change or add data to single model components, even during construction phase. These are intended deviations from the design. For instance, a monitoring unit is added in order to capture the energy consumption of motors that operate the window shadings, or a second layer of water sealing has to be applied, because of unexpected soil properties. Such "on the fly" changes are only possible if single components in the model may be edited without affecting other components.

Feature 3 Storing data
Real-time display of monitoring data is usually not enough, measured data has to be appropriately and safely stored. Ideally, storage occurs in perfect match with the digital model, even if it is in a separate database. In this way, potential loss of information is easy to detect and to recover from.

Feature 4 Incorporating automation logic into model components
The id of the monitoring unit in a data record must indicate the loads to which the data, measured at the respective time, refers. In addition, it should be possible to identify a monitoring unit that monitors any given component, e.g., a ventilation unit, whose electricity consumption needs to be assessed. In addition, the functionality of automation elements should be formalized in a standardized language.

Feature 5 Component templates
When adding components to the digital model, it should be clearly indicated if they are relevant for monitoring during the operational phase. In order to avoid a planner having to remember whether the component is being monitored or not, observation component templates that have already incorporated such information should be readily available.

Feature 6 Visualization of monitored data in a 3D CAD model
It should be possible to visualize monitoring units and their measuring scope, i.e., all zones and components measured by this unit. Moreover, visualization of monitoring data should be possible. For instance, if room temperature can be measured by room, the values should be visualized by room.

Feature 7 Role and right management
The role and right management should be clear for all components. Thus, it is clear which stakeholder is responsible for which parts.

3. Simultan from Design through Construction into Operation

In this section, we present our approach - the data model SIMULTAN, and provide arguments for its suitability for supporting the building life cycle from the design into the operation phase. In SIMULTAN each stakeholder, who is involved in the planning or construction process, such as architect, civil engineer, HVAC engineer or electric engineer has an individual view on the data model. In addition, each stakeholder has specific read and write rights for each component of the building model. This concept supports **Feature 7**. A more detailed discussion of managing roles and rights can be found in (Paskaleva et al, 2019). For developing the digital model of buildings there are two views onto the data model: (i) geometry 3D model view and (ii) list-style backend view that shows all model components and their parameters and calculations. The second view also includes components that have no immediate representation in 3D, e.g., wall constructions or materials.

The data model supports the creation of digital models starting with the design phase (Paskaleva et al, 2019). In the following, we discuss the parts of SIMULTAN that realize further features – those presented in Section 2.

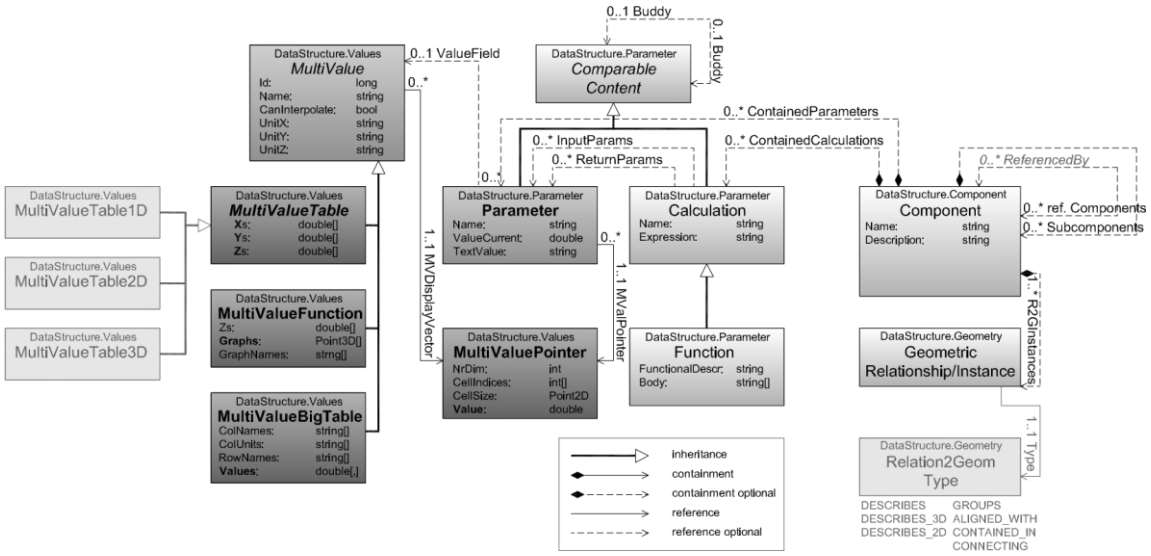

Figure 1: Data model excerpt illustrating the representation of large numeric data. Each Parameter can reference a MultiValue object, representing a scalar field, a graph filed or a 2d table, and access a specific value by means of a pointer (see type MultiValuePointer).

In order to manage simulation or monitoring data, the data model must incorporate appropriate containers – these are the sub-types of *MultiValue* – an abstract container for multiple numeric values. These values can be organized in scalar fields (*MultiValueTable*), graph fields (*MultiValueFunction*) or regular two-dimensional tables (*MultiValueBigTable*). Figure 1 shows those types and their interconnection. The scalar fields are regular cell grids in one, two or three dimensions (*MultiValueTable1D, MultiValueTable2D* and *MultiValueTable3D* respectively) with each cell boundary being associated with a value. The values within the cell can be determined by applying a user-defined interpolation method – e.g., nearest neighbor or linear interpolation. This type of container can manage the results of a computational fluid dynamics simulation. Graph fields, on the other hand, contain multiple named one-dimensional function graphs – those can represent, e.g., the characteristic of a ventilator depending on rotation speed. Finally, two-dimensional tables can contain any type of numeric value depending on two variables – including time series produced by a simulation or gathered by a monitoring system. The two-dimensional tables can be incorporated in matrix and vector calculations, which allows the simultaneous processing of large data sets – e.g., the values for all time stamps in a time series. This is possible via the type *Calculation* (see Figure 1), which references input and return parameters. If at least one of the input parameters references a 2d table as its ValueField, the calculation switches automatically from scalar to vector mode. Chaining vector calculations allows the definition of data processing algorithms that can be called, for example, any time the input data changes. Such algorithms can be incorporated in component templates (see Figure 3).

It is possible to store the data produced by simulations as well as delivered by monitoring systems on every level of the data model. Consequently, it is quite straightforward to compare information and to draw conclusions out of this comparison. Thus, the structure provided by the *MultiValue* type supports **Feature 1** and **Feature 3**. In addition, the results are linked to the geometry 3D model view, since our two views on the data model are synchronized with each other (Paskaleva et al, 2019). This enables the data model to fulfill **Feature 6** as well.

Furthermore, for storing and managing data within the digital building model (**Feature 3**) our approach also supports attaching relevant information to the already established component structure as an asset via the ReferencedAssets relationship (see Figure 2). This structure is similar to the *IfcProxy* or *IfcExternalInformation* entities in IFC (buildingSMART, 2018). However, unlike in IFC,

the data in a *GeometricAsset* or *DocumentAsset* is not saved as part of the structure – it resides in a linked File that can be addressed and opened via its parent component.

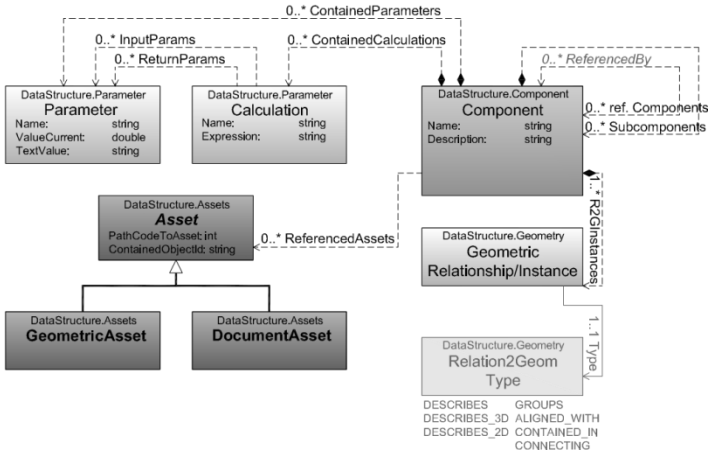

Figure 2: Data model excerpt illustrating the incorporation of any type of additional information into the data model. The Asset type, along with its sub-types GeometricAsset and DocumentAsset, enables the attachment of arbitrary files to any component – including product specifications or photographic documentation of building progress.

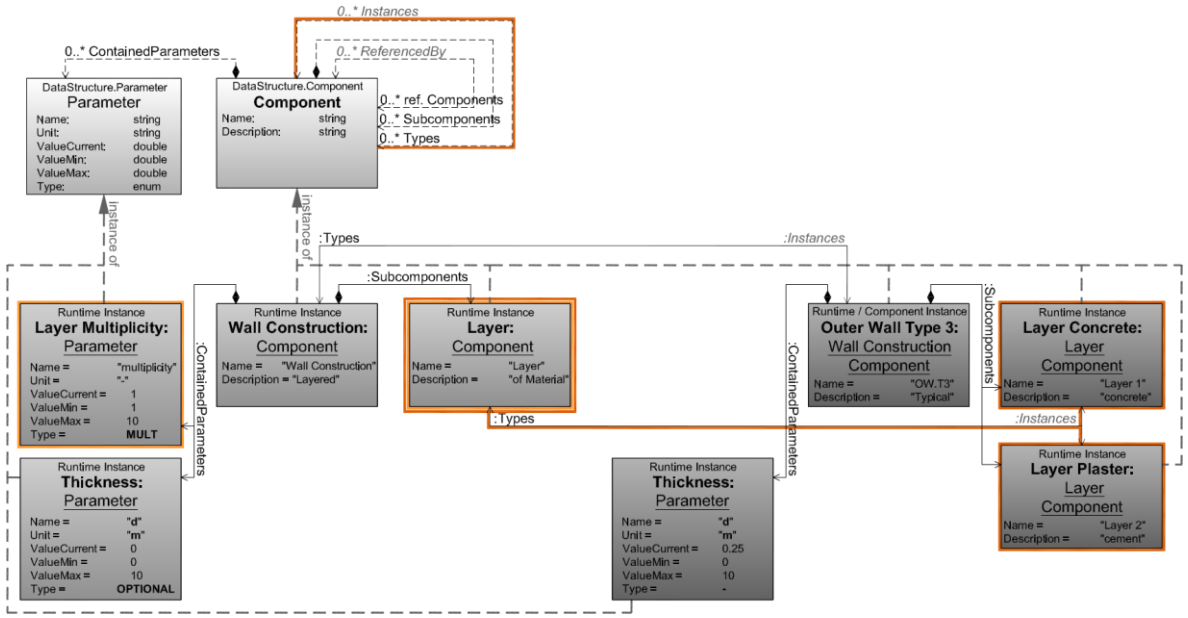

Figure 3: Using components as templates. The top part shows the types Component and Parameter. The bottom – the objects representing part of the model of a building at runtime (e.g., during manipulation in a software tool). The template for a wall construction is displayed on the left. It contains one sub-component Layer and an associated parameter Layer Multiplicity that defines the admissible number of Layer sub-components (1 to 10). It also contains an optional parameter Thickness. A component using Wall Construction as a template is shown on the right – the Outer Wall Type 3 component. It contains two sub-components using the Layer Component as a template (follow the orange lines) – Layer Concrete and Layer Plaster. It also contains the parameter Thickness and assigns the value 0.25 to it.

Once the data structure for a specific purpose (e.g., a wall construction) is built and tested, it can be used both as a requirement and as template for the generation of further components. The top part of Figure 3 shows the *Component* type with the reflective relationship Types highlighted in orange. This relationship allows the simulation of instantiation at run-time, or, in other words, using one *Component* instance (e.g., Layer) as template for another *Component* instance (Layer Concrete and Layer Plaster).

The grey dashed lines in Figure 3 indicate the instantiation performed by the inbuilt type system of a programming language with object oriented design capabilities (e.g., Java or C#). The model of a

building contains only run-time instances (the bottom part of Figure 3) – objects whose type is defined by a class (e.g., *Component*). In our case, this is a linguistic or syntactic type. It helps us establish a structure, but lacks semantics or meaning. Similar to the type-object pattern in object-oriented design, the meaning can be provided by a template – e.g., the *Wall Construction* component instance. It defines a sub-component *Layer* and a parameter *Layer Multiplicity* that sets the admissible number of *Layer* sub-components – e.g., ValueMin = 1 and ValueMax = 10, or between 1 and 10. It also defines an optional parameter *Thickness* with name "d" and unit "m". The bottom right part of Figure 3 shows the *Component* instance *Outer Wall Type 3*, which uses *Wall Construction* as a template. It contains an admissible number of sub-components of type *Layer Component* (two) as well as the optional parameter *Thickness*.

The use of a template component guarantees that all components conforming to it, i.e. containing all of its mandatory elements in admissible quantities, are compatible with each other, and can exchange or aggregate information without distortion or loss. In other words, conforming to a template enables an automated check for requirement fulfillment, while simultaneously preserving a degree of design freedom by allowing the definition of additional elements. Thus, the data model supports **Feature 2** and **Feature 5**.

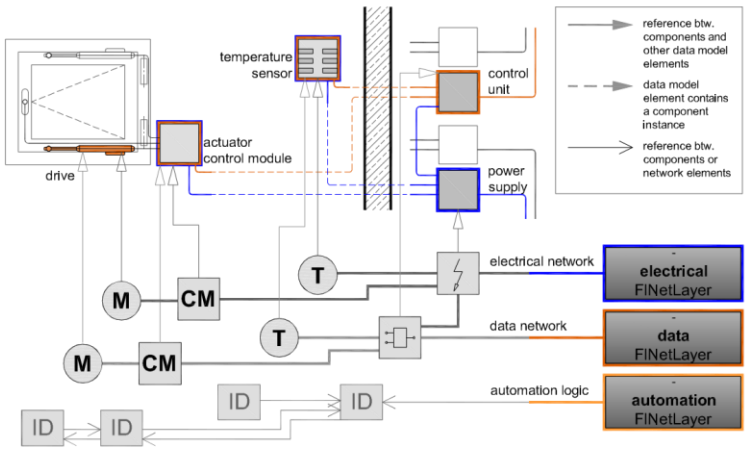

Figure 4: The data model elements allowing the definition of building service networks coupled with building automation. In this case there are three networks involved: two physical - the electrical (in blue) and the data or information network (in orange), as well as one logical – the building automation logic (in yellow). Each network resides on a dedicated network layer, which manages both visibility and access to the network.

Figure 4 shows a small example of the building automation system, i.e. a window that is opened or closed automatically depending on the room temperature. Depicted are the drives that operate the window, the actuator control module that triggers these drives, the room temperature sensor and the room control unit and power supply. There are two physical networks involved – the electrical and the data networks, and one logical network – the automation logic. Each of those networks can be managed as a separate scheme placed on its own layer that manages both its visibility and access (see also the relationship between types *FlNetElement* and *FlNetLayer* in Figure 6).

The networks in Figure 4, however, share multiple elements. In order to guarantee the consistency between them we organize them in groups (depicted in Figure 5) via the relationship between types *FlNetElement* and *FlNetElementGroup* shown also in Figure 6. This grouping ensures that all instances of *FlNetElement* in the same group contain the same instance of the same component (see relationship Content to *Component*) - for more details see also (Paskaleva et al, 2019).

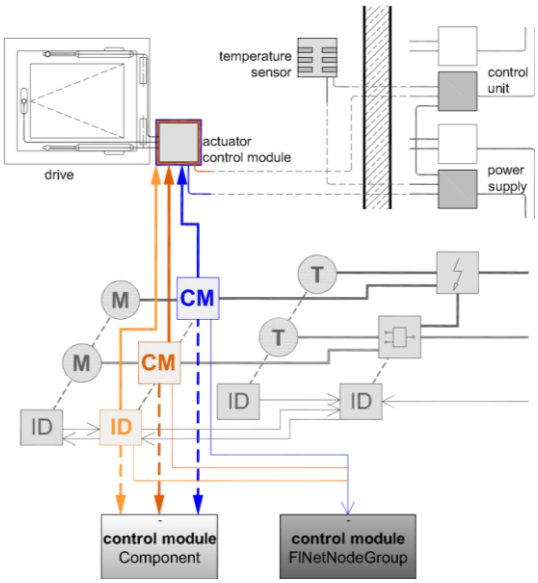

Figure 5: Managing consistency across multiple networks on different layers and in the area of responsibility of different stakeholders. Network elements that reference the same physical entity but model different aspects of its functionality, are organized in groups – in this case the three nodes representing the actuator control module are grouped in the "control module" FlNetElementGroup and, therefore, aware of each other.

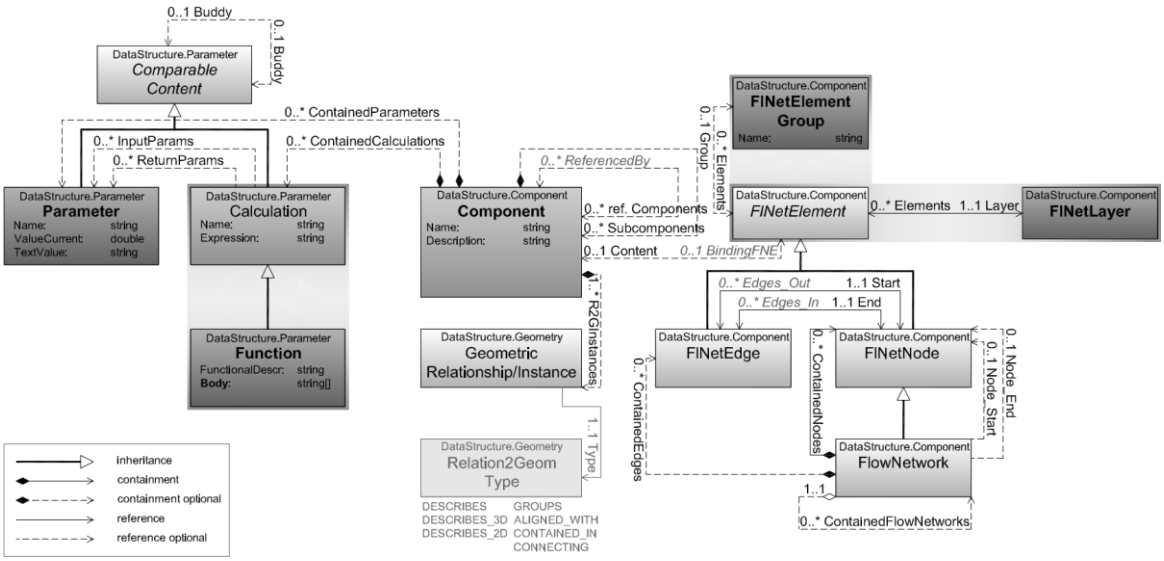

Figure 6: The data model excerpt illustrating the management of heterogeneous networks. All network elements (of type FlNetElement and it sub-types) can be organized along two orthogonal axes – one along the type FlNetLayer that separates areas of responsibility, the second – along type FlNetElementGroup that ensures consistency between elements representing the same physical entity in different networks. All elements in a group reference the same Component instance via the Content relationship.

The building automation logic is based on the requirements for (i) unique identification of each element, (ii) element connectivity, and (iii) functional description of control units – see (Kranz, 2013). The network infrastructure provided by the types *Component*, *FlNetElement* and its subtypes satisfies the first two requirements. In order to satisfy the third we provide a new sub-type of *Calculation* – *Function* (left-hand side of Figure 6). Along with the input and return parameters, *Function* also contains not just a symbolic expression but a function body definition (the algorithm to be executed), similar to that found in most programming languages.

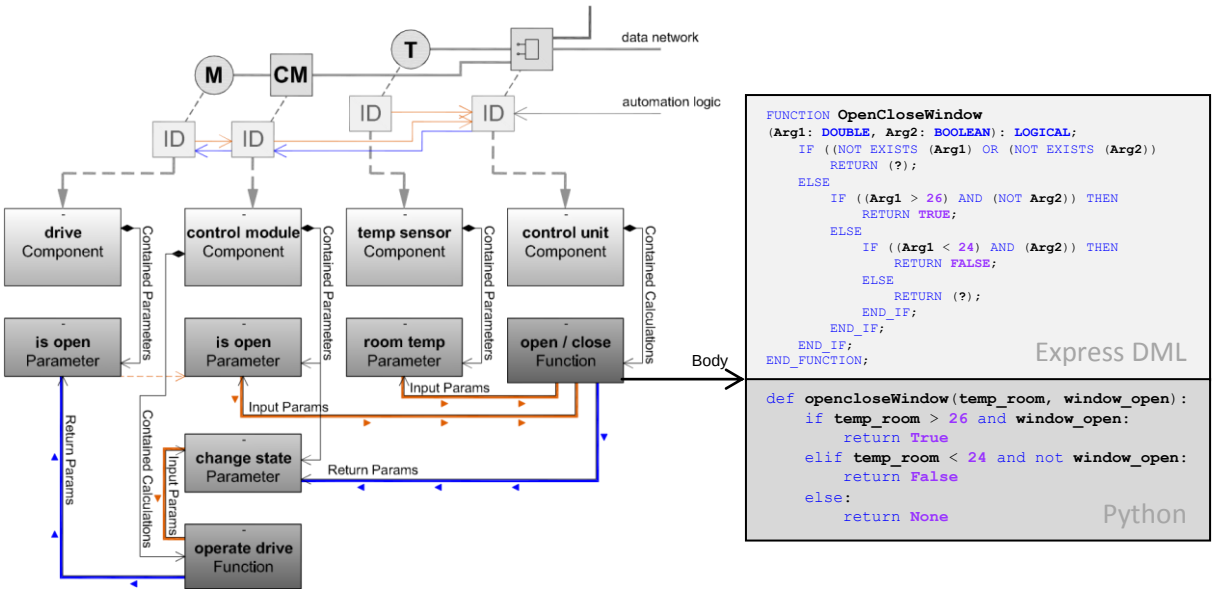

Figure 7 Building automation logic of the model presented in Figure 4 and Figure 5. Each of the automation nodes contains the instance of a Component. For example, the control unit node contains an instance of a component that includes the function "open / close" (last on the right). It takes input from the instances in the temperature sensor and the window control module (in orange) and delivers output to the control module (in blue), which opens or closes the window. The body of the function "open / close" uses, as an example, the syntax of the Express DML (ISO 10303-11:2004), and, directly below, the syntax of Python.

Figure 7 shows an excerpt from the model we described above, which focuses on the automation logic. Each node, carrying a unique id, has a *Component* instance associated with it – for the control unit, the temperature sensor, the window control module and the drive. The parameters of the component instances can be connected via the network (dashed orange line) as well as via the functions (continuous orange and blue lines) – delivering input and output information. The functionality of an element, which is generally documented as a natural language description during the planning phase, can be formalized in a number of programming or data modeling languages that are already in use. Figure 7 shows on the right hand side an example of the Express Data Modeling Language (DML) syntax (ISO10303-11:1994, 1994), which is used by IFC open BIM standard, and, directly below it, the same logic in Python. The algorithm is defined as follows: *If the input is valid and the temperature of the room is above 26° C and the window is closed, open it. If the input is valid and the temperature of the room is below 24° C and the window is open, close it. Otherwise do nothing*. For the sake of simplicity, it does not consider the temperature of the outside air. This aspect of our data model is a first step towards fulfilling **Feature 4**.

4. Proof of Concept

Since 2017, we have been working on a research and development project with the Vienna airport (commissioner) involving more than 100 existing and future buildings, according to the airport's strategic development plan. These buildings as well as the heat and cooling networks and the electricity grid connecting the buildings are modelled with the data model described in this paper. So far, four of the largest objects (comprising 100.000 m²) have been built as digital models: an office building and three airport terminals, one of them including shopping and gastro zones. Dynamic thermal simulations have been carried out during the design phase, in order to discover potential optimization potentials. After the completion of all four buildings (ca. by 2021) each will have its digital twin allowing, e.g., to predict the building's performance based on weather forecasts. In addition, a monitoring system covering many of the 100 existing buildings will be established. This system will enable a detailed comparison of simulation data and real performance.

As part of the expert support provided by TU Wien, based on our digital model, we already identified two performance tracking errors: (i) In one building heating and cooling occurred simultaneously due to a faulty control valve. (ii) An electricity meter that was to be read out manually was read out a few days too late, which could be easily discerned by analyzing the corresponding data.

5. Conclusion and Outlook

The SIMULTAN data model maintains the separation of responsibilities at various levels of detail, as well as data consistency across multiple domains. It allows not just the modeling of buildings but also of requirements in the form of templates, and of automation specifications in the form of logical networks and formal algorithm descriptions. It enables the comparison of simulation and monitoring data and, therefore, provides the basis for tracking the relationship between target and actual values during the operation phase. Future developments include the integration of the formal algorithm descriptions into simulations by providing "on the fly" code compilation, and the graphical 3D visualization of parameter values according to user-defined rules.

Acknowledgements

Wiener Stadtwerke, TU Wien URBEM (https://urbem.tuwien.ac.at/home/EN/), Austrian Ministry for Transport, Innovation and Technology (bmvit) and Vienna International Airport have supported this work.

References

Borrmann, A., König, M., Koch, C. and Beetz J. (2015) *Building Information Modeling. Technologische Grundlagen und industrielle Praxis*. Wiesbaden: Springer Vieweg.

buildingSMART (2018) http://www.buildingsmart-tech.org/specifications/ifc-overview, last access: 08-Jan-2019

Goedert, J. D. and Meadati P. (2008) *Integrating Construction Process Documentation into Building Information Modeling*. Journal of Construction Engineering and Management, Volume 134 (7), pp. 509-516

Hens, H. (2012) *Performance based building design*. Wiley, Ernst&Sohn.

ISO 10303-11:1994 (1994) *Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual*

Kranz, H. (2013) *BACnet Gebäudeautomation 1.12: Grundlagen in deutscher Sprache*. CCIBuch (in German)

O'Sullivan, D.T.J., Keane, M.M., Kelliher, D. and Hitchcock, R.J. (2004) *Improving building operation by tracking performance metrics throughout the building lifecycle (BLC)*. Energy and Buildings, Volume 36 (11), pp. 1075-1090.

Paskaleva, G., Wolny, S. and Bednar T. (2018). *Big-open-real-BIM Data Model - Proof of Concept*. In Proceedings of the 7th International Building Physics Conference, IBPC2018, Syracuse, NY, USA. pp. 1083 - 1088.

Paskaleva, G., Lewis, T., Wolny, S., Steiner, B. and Bednar T. (2019). *SIMULTAN as a Big-Open-Real-BIM Data Model -Proof of Concept for the Design Phase*. CIB World Building Congress, WBC 2019.