Quality Check of openBIM Input Data for Thermal Building Simulation based on mvdXML

Andreas Geiger¹, Karl-Heinz Häfele¹, Veit Hagenmeyer¹
¹Karlsruhe Institute of Technology, Institute for Automation and Applied Informatics, Karlsruhe, Germany

Abstract
Success and quality of thermal building simulations strongly depend on consistency, correctness, and completeness of BIM data. Therefore, it is necessary to check in this context the content of openBIM data before transferring it into a respective simulation tool. With the buildingSMART standard Model View Definition XML (mvdXML), exchange scenarios can be formalized which are testable with software tools.

In the present paper, the current capabilities of mvdXML rules are analyzed and a first approach of Model View Definition (MVD) rule sets to cover exchange requirements for thermal building simulations is introduced, including formalizing the MVD rule sets as mvdXML. Using an additional module of the software application FZKViewer, the defined rule sets are validated and tested in an openBIM workflow for thermal building simulation.

Introduction
Today, Building Information Modelling (BIM) covers many domains of the building life cycle. As buildings and activities related to buildings consume a major proportion of the total energy worldwide (UNEP, 2019), the domain of Building Performance Simulation (BPS) has gained significance in the building life cycle. Using building information models for simulations reduces preparing time and increases quality of results. Furthermore it is possible to feed the simulation results back into the BIM.

However, success and quality of building thermal simulations strongly depend on consistency, correctness, and completeness of available BIM data. Therefore, it is necessary to validate BIM data before using it in thermal building simulation.

With the Industry Foundation Classes (IFC), buildingSMART International (bSI) published an open and non-proprietary data model (openBIM) for storing and exchanging building information models as an international standard (ISO 16739, 2018). As IFC is intended to be a holistic model that supports the complete building life cycle, different domains and applications usually use well-defined subsets of the model, so-called Model View Definitions (MVD). In order to formalize MVDs, buildingSMART released Model View Definition XML (mvdXML) as a bSI Standard (Chipman, 2016).

The quality checks performed on IFC models can be classified in three categories:

- Consistency (verification of syntax and semantics),
- Correctness (verification of geometry and non-geometric entities),
- Completeness (verification of model content).

The verification of the syntax and semantics of an IFC document is straightforward with available software (IFCCheckingTool, 2019).

Furthermore, the basic verification of geometry is usually carried out and reported while importing the IFC model into the target system. Some applications offer extended functions that find and visualize geometrical errors, such as non-planar faces or erroneous surface orientation. In addition, many applications offer clash detection to detect geometrical issues even when base geometry is correct.

Moreover, verification of model content, also called code checking, can be performed in two different ways:

1. Using a proprietary format for the definition and description of content and corresponding rules,
2. Using an open and standardized format for the definition and description of content and corresponding rules.

In the first case, both commercial and academic solutions are available. The disadvantage of this solutions is that rules must be redefined in the case of each application.

In the second case, both the BIM data (IFC) and the data requirements (mvdXML) are interoperable, owing to open and neutral standards. Software implementations in this application area are mainly developed in the academic field.

In this context, the present paper focuses on checking completeness of openBIM data (IFC) with mvdXML. Even if the described approach is suitable for checking exchange requirements in general, the target application in this paper is to control data exchange between Computer Aided Architectural Design (CAAD) and BPS. The aim is to detect flawed and missing properties, to find inconsistencies in the data set and to ensure a correct spatial subdivision of the building. By performing this quality check, the suitability of the BIM data sets can be evaluated before running time consuming simulations.

The first section of this paper gives an overview of the general use of mvdXML including available checking tools. The next two sections introduce the basics of mvdXML and describe future extensions of mvdXML. Following the section providing the details of the
specified mvdXML rules, the latest in-house software implementation for performing quality checks are explained in a further section.

The approach used in the definition of mvdXML in this paper does not follow the top-down methodology of buildingSMART. Instead, the exchange requirements are defined and encoded in mvdXML directly (bSI IDM, 2010).

openBIM quality check with mvdXML

“A Model View Definition (MVD) is a subset of the overall IFC schema to describe data exchange for a specific use or workflow, narrowing the scope depending on the needs of the receiver” (bSI MVD, 2020). This means that a MVD can exclude entities, properties, and relations for a specific use case, but can also force them to be mandatory for the use case. Complex rules can be defined by a logical combination of basic rules.

An MVD can be generated in different ways:
- a documentation describing model content and additional exchange requirement,
- using authoring tools (bSI IfcDoc, 2020; BIMQ, 2021) for a more structured way to define exchange requirements,
- using mvdXML as a formal, computer-interpretable description of MVDs.

The preferable way to create MVDs is using an authoring tool (bSI IfcDoc, 2020) which supports both, the documentation of the MVD and the formalization with mvdXML.

MVDs cover individual, company / organization wide or general data exchange requirements. buildingSMART lists a number of MVDs on the website (bSI MVD Database, 2020). Only the Reference View of IFC4 ADD2 TC1 is available with both an online documentation and the corresponding mvdXML file for downloading. This buildingSMART list contains a so-called “Energy Analysis View” labelled as draft, but no further documentation or downloads are available. Space boundaries are essential for BPS and it should be mentioned that for IFC2x3 a “Space Boundary Add-On View” was defined in 2010 (bSI SB Add-On View, 2010). As the space boundaries concept is extended in IFC4, this add-on view must be updated and adapted for IFC4 models.

Beside the MVDs listed by bSI, there are several activities in the official and academic area to define MVDs for information exchange between BIM and thermal building simulation applications.

In the European Project HESMOS, data requirements were defined for an Extensible Energy-enhanced BIM framework (eeBIM). These definitions cover several use cases, like BIM to Energy, Energy Simulation Results and Energy Issue Reports. The requirements for each use case are listed in a neutral, descriptive way. Only for the use case BIM to Energy a detailed table of exchange requirements is given, which can be the basis for an MVD (Liebich, 2011). A complete MVD including the formalization in mvdXML seems not available.

The Platform for District Energy-Efficient Retrofitting Design of the European Project OptEEmal (OptEEmal, 2019) contains an IFC data workflow which includes a checking tool for mvdXML. The mvdXML rule sets for checking the space boundaries and various property sets are generated using the buildingSMART IfcDoc tool (Katsigarakis, 2019). The applied mvdXML rule sets seems not publicly available.

The “Design to Building Energy Analysis” MVD (“Design to BEA” MVD) was released as an US National BIM Standard (NBIMS, 2015). This MVD is developed for IFC2x3 and documented in an Information Delivery Manual (IDM) including detailed interactive diagrams (Design-to-BEA, 2011). For this MVD a mvdXML formalization seems not available, as well.

All previous mentioned MVDs are mainly focusing on architectural aspects like building structure (location, orientation), building envelope (space boundaries), space types (function, thermal requirements) and the thermal characteristic of the building elements (materials). Less consideration is given to the building services systems, including the topology of such systems (e.g., building heating system).

In order to fill this gap, the International Energy Agency IEA EBC Annex 60 project focuses on a MVD which significantly extends the consideration of building services. It includes the system components (boiler, pipe, radiator), the topology of the system (connection port), schedules (time series) and controls for components. The MVD is created by using the IfcDoc tool (bSI IfcDoc, 2020) developed by bSI. The MVD documentation and the possible mvdXML seems not to be publicly available (Pinheiro, 2018).

Provided a valid mvdXML file, the content of an openBIM model can be checked automatically. The openBIM model check can be performed in two ways:
- by including the model check in an integrated BIM workflow, possibly based on BIM server technologies,
- by using standalone tools for model checking, allowing file-based workflows.

While the first way requires certain IT infrastructure resources, the second way usually can be performed easily on available desktop hardware. In the following, two available standalone solutions for openBIM model checking with mvdXML are discussed:

Simplebim is a commercial application to process IFC models. It offers multiple functionalities such as enriching, cleaning, merging and splitting IFC models. In addition, Simplebim provides an add-on for model checks based on mvdXML Version 1.1. Both, Simplebim and the add-on can be tested for 30 days free of charge (Simplebim, 2021). After loading the IFC model, the applicable mvdXML file can be imported, native Simplebim templates can be generated and the model can be tested. For the verification of the test results all tested IFC entities are highlighted in the validation window and marked as valid or not. A separate window shows detailed results for selected IFC entities. Simplebim has not
implemented the complete mvdXML functionalities and thereby has some limitations. Not all checks specified in the present paper in the sequel could be performed with Simplebim.

Furthermore, the xBIM toolkit is an open-source project to support openBIM applications. Beside the source code, the xBIM Xplorer can be downloaded as a ready to use application (xBIM Xplorer, 2021). With a Plugin the mvdXML import, export or mvdXML. The mvdXML format was released in 2013. IFC inheritance are described attributes and relationships of a technical requirements for software vendors to meet the needs of construction project know which IFC components. With this approach linked and supports the information 2005 information about a entity in the graphic window and in the project hierarchy. During the lifecycle of a facility processes, provides specific demands, their requirements for software vendors to meet the needs of all stakeholders. An IDM represents standardized template documents that are linked by an MVD to specific IFC classes and attributes. With this approach, all stakeholders involved in a construction project know which IFC components meet their needs. At the same time, an IDM describes the requirements for software vendors to meet the needs of their customers. Based on the demand of an IDM, the technical requirements specification MVD is generated. It contains all entities, attributes, properties, and relationships of a specific BIM use case. Relevant attributes and the relations between entities based on the IFC inheritance are described using concepts (IUG, 2012; Karlshoej, 2012).

![Figure 1: Schematic diagram of the mvdXML format](https://doi.org/10.26868/25222708.2021.30501)

As part of the IFC version 4 development, the mvdXML format was released in 2013. It allows a neutral and computer-interpretable representation of MVDs. Due to limitations of the first version regarding validation rules, applying them to an MVD and selected Exchange Requirements, version 1.1 was released in 2016 with an enhanced validation part (Chipman, 2016).

The core structure of mvdXML is divided into two major sections. The first section is called Templates and contains reusable ConceptTemplates representing a sub-graph of the IFC schema (Figure 1 right). The second section is called Views and specifies how the templates should be applied within a specific MVD and Exchange-Requirement (Figure 1 left).

With the element ModelView an entire MVD can be represented. This element contains all definitions of an MDV, is related to a specific IFC schema and must have the elements ExchangeRequirement und ConceptRoot. With an ExchangeRequirement the context of a Concept can be defined, for example whether a Concept is relevant for import, export or for both. The ConceptRoot is the base of a rule set definition assigned to a so-called root entity. A root entity usually describes an entity derived from IfcRoot representing spatial or building elements. In addition, the Applicability element can be used to define further conditions that must be fulfilled by a ConceptRoot. These conditions are used as a filter. For example, a condition can be defined for walls that should be exclusively applied to load-bearing walls.

For defining rules within a Concept, the IFC classes and attributes are not directly used. Instead, they refer to ConceptTemplates which can be seen as prototypes and are reusable. This concept offers a universal and flexible way for the definition of rules. The individual rules are assigned to a Concept via the element TemplateRules. This allows the definition of a hierarchical tree of logical expressions, where logical operators combine individual rules.

Each expression of a TemplateRule has a specific syntax and includes the components parameter, metric, operator and value in the following format:

```
{Parameter} {Metric} {Operator} {Value}
```

The Parameter represents a keyword defined in the related ConceptTemplate and refers to a specific path within the IFC schema. The Metric indicates how the Parameter should be used while the Operator defines how to use the Value. The allowed Values for the Metric are shown in Table 1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Specifies the value of an attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Specifies the size of an aggregation or the value of an integer</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the value assigned to the attribute</td>
</tr>
<tr>
<td>Unique</td>
<td>Specifies a value must be unique</td>
</tr>
<tr>
<td>Exists</td>
<td>Specifies the existence of a value</td>
</tr>
</tbody>
</table>

In Figure 2, a simple example of a Concept containing Applicability and TemplateRule is given. This Concept is applied to all instances derived from the type IfcProduct. In Applicability, it is additionally specified that the rule should not be applied to Entities of the type IfcAnnotation.
and IfcVirtualElement. The rule itself refers to a ConceptTemplate, where “Name” is defined for the attribute IfcProduct.Name. This attribute must be instantiated to satisfy the rule. In this example, extended functionality (SELF) is being used that is not part of mvdXML 1.1. With the use of the new parameter SELF, a filter can be easily defined so that the rule is not applied to instances of the type IfcAnnotation and IfcVirtualElement. This mechanism has been adopted from EXPRESS (ISO 10303-11, 2004) and is available in many object-oriented programming languages to refer to the current instance.

As this example shows, there are some rules which can either be not formulated at all or are cumbersome with the current version 1.1 of mvdXML. Therefore, the rules introduced in the present paper are based on a draft version of mvdXML 1.2.

{ }
<Applicability>

and

SELF[Type]=’IfcAnnotation; IfcVirtualElement’

Name exists

Name[Exists]=TRUE

Figure 2: Concept with Applicability and TemplateRule

Proposed extensions for mvdXML

This proposed extensions are currently being developed within a group of experts from buildingSMART Modelling Support Group (MSG). The latest version is available on the official buildingSMART on GitHub (bSI mvdXML, 2020). Proposals are discussed on the buildingSMART forum and appropriate feedback is taken into account. The final proposal will be submitted to buildingSMART International for an official update of the mvdXML standard.

Two extensions are presented in the context of this contribution. The first is a possibility to set the number of instances. This gives the user the ability to specify whether and how many instances of an entity are permitted. The adaptation is done by two new attributes minOccurrence and maxOccurrence at the element Applicability (see Figure 3).

Figure 3: Extension of the element Applicability

The second extension allows the definition of user-defined messages. The result of an mvdXML rule check normally returns either successful or failed. The message text is automatically generated by the particular testing software from the context of the rule test. This messages are usually difficult to interpret by end users, therefore a concept for user-defined messages is proposed, the new element RuleMessage (see Figure 4). On the one hand, this enables messages that are easier to understand for the end-users, and on the other hand, it also gives the ability to specify different message types (success, failure, warning or comment).

Figure 4: New proposed element for end-user messages

mvdXML Rules

As described, there are different categories for evaluating the quality of openBIM data. In the following, it is assumed that the data is syntactically and semantically correct to the corresponding IFC schema (e.g., IFC2x3, IFC4).

The presented approach for defining and creating rules does not correspond to the usual buildingSMART top-down methodology creating an IDM, generating MVD and formalizing it using mvdXML. Instead, a solution-oriented bottom-up approach is applied, in order to formulate the requirements of the thermal building simulation. This is preceded by an analysis of existing openBIM building models that identifies problems in real-life examples, which leads to Table 2 containing the relevant elements and their attributes, representing the minimum requirements for an openBIM building model used for thermal building simulation.

Table 2: Formal definition of checking rules

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper project structure is required</td>
<td>Site, Building, Building storey</td>
</tr>
<tr>
<td>Building has to be georeferenced by</td>
<td>Site – Latitude / Longitude</td>
</tr>
<tr>
<td></td>
<td>Map conversion – projected SRS</td>
</tr>
<tr>
<td>Building (minimum number of building elements per storey)</td>
<td>1 roof</td>
</tr>
<tr>
<td></td>
<td>4 walls / 1 façade / 4 curtain walls</td>
</tr>
<tr>
<td></td>
<td>1 base slab</td>
</tr>
<tr>
<td>Building elements are mandatory, containing</td>
<td>Geometrical representation</td>
</tr>
<tr>
<td></td>
<td>Space boundary</td>
</tr>
<tr>
<td></td>
<td>Material definition</td>
</tr>
<tr>
<td>Material specifications are required with:</td>
<td>Material name</td>
</tr>
<tr>
<td></td>
<td>Material layer</td>
</tr>
<tr>
<td></td>
<td>Physical material properties</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
At least one space per storey has to be defined

- Space geometry

Space boundaries are needed (1\textsuperscript{st} or 2\textsuperscript{nd} level)
- Standard space boundaries (IFC2X3)
- Internal / external boundary
- Virtual or physical element
- Related to space
- Extended space boundaries (IFC4)
- Internal / external
- Virtual or physical element
- Related element
- Related to space or spatial element
- Inner boundaries \(\rightarrow\) parent boundaries
- External boundary condition
- Corresponding boundary (2nd level)
- Related element
  - IfcOpeningElement \(\rightarrow\) virtual
  - IfcBuildingElementProxy not allowed

Spatial elements should be defined
- External boundary conditions

A zone can be defined (thermal zone)
- Space \(\rightarrow\) Zone 1.1
- Temperature Zone \(\rightarrow\) Temperature Space
- Only spaces with same conditions (Temperature)

The rules introduced include checks for attributes - such as if the attribute \(\text{Object.Name}\) is defined or has a specific value - and checks for properties - e.g., the property \(\text{ThermalTransmittance}\) of a common property set should have a value and a specific valuation. These simple types of rules are the basis of the more complex rules shown in the present paper, and are not discussed further.

The basic prerequisite is that a "meaningful building model" is available. This rule is not easily formulated. For this, it is assumed that a building has at least 4 walls, a roof and a floor slab. If this is not the case, a warning is given, as there may be modelling variants with facade elements, that represent a building correctly. To realize this, the mechanism for user-defined messages described below is used here. This check is necessary as some models used for thermal building simulation look visually correct, but do not have a semantically correct building model. This can happen in CAAD tools either by incorrect mapping of the internal building elements to IFC elements or by the export settings. For example, all components being exported as \(\text{IfcBuildingElementProxy}\).

Besides ensuring that a building is instantiated with proper building elements, it must be guaranteed that the building elements contain the relevant attributes and properties to perform a thermal building simulation. Therefore, two more complex rules are discussed in more details. These are a check for the relevant physical parameters of the individual building elements and a check for the presence of space boundaries. Both rule sets are presented in a simplified way, highlighting the key conditions.

\[
x \in \text{IfcBuildingElement} \mid ((\text{A}(x) \text{ and B}(x) \text{ and C}(x)) \text{ or } D(x))
\]

A = \{ \text{PropertySetName}[\text{Value}] = \text{Pset_MaterialThermal} \text{ and } \text{NominalValue}[\text{Value}] > 0.0 \} \tag{1}

B = \{ \text{PropertySetName}[\text{Value}] = \text{Pset_MaterialThermal} \text{ and } \text{SimplePropertyName}[\text{Value}] = \text{SpecificHeatCapacity} \text{ and } \text{NominalValue}[\text{Value}] > 0.0 \} \tag{2}

C = \{ \text{PropertySetName}[\text{Value}] = \text{Pset_MaterialCommon} \text{ and } \text{SimplePropertyName}[\text{Value}] = \text{MassDensity} \text{ and } \text{NominalValue}[\text{Value}] > 0.0 \} \tag{3}

D = \{ \text{PropertySetName}[\text{Value}] = \text{reg_Pset_\text{"([A-Z][a-z])Common"}} \text{ and } \text{SimplePropertyName}[\text{Value}] = \text{ThermalTransmittance} \text{ and } \text{NominalValue}[\text{Value}] > 0.0 \} \tag{4}

In the first rule set (1), it must either be ensured that a building element has a layered structure with layer thickness. Further, the physical parameters thermal conductivity (2), specific heat capacity (3) and mass density (4) with values must be assigned to each individual layer. If this is not the case, or if the component (e.g., window, door, ...) does not have any layers, the value of the corresponding common property set (e.g., \text{Pset_WindowCommon}) must be set instead (5).

Beside material information, space boundaries are an important component for BSP. These boundaries represent the connecting surfaces between building elements and the adjacent spaces. They are relevant for heat conduction and heat storage processes of building elements. In IFC, space boundary information is represented by the relation \text{IfcRelSpaceBoundary}. This relation forms the connection between a space and the adjoining building elements. A distinction is made between 1\textsuperscript{st} level and 2\textsuperscript{nd} level. 1\textsuperscript{st} level means the entire adjacent area as seen from space. 2\textsuperscript{nd} level takes into account all adjacent building elements and space on the other side of the building element and divides the surface accordingly.

\[
x \in \text{IfcRelBuildingElementProxy} \mid ((\text{A}(x) \text{ and B}(x) \text{ and C}(x)) \text{ or } D(x))
\]

A = \{ \text{SBRelBuildingElement}[\text{Exists}] = \text{TRUE} \} \tag{5}

Figure 5 shows - based on the example of the KIT Musterhaus, part of the research project EnergyLab 2.0 (Hagenmeyer, 2016) - the architectural model on the left and the model of the space boundaries on the right. With IFC4 these relations have been extended to include specific entities for 1\textsuperscript{st} level (\text{IfcRelSpaceBoundary1stLevel}) and 2\textsuperscript{nd} level (\text{IfcRelSpaceBoundary2ndLevel}) boundaries. In addition, IFC4 now also allows the definition of the outer building envelope with the corresponding external boundary conditions (such as air, water or earth).

\[
x \in \text{IfcRelBuildingElement} \mid ((\text{A}(x) \text{ and B}(x) \text{ and C}(x)) \text{ or } D(x))
\]

A = \{ \text{SBRelBuildingElement}[\text{Exists}] = \text{TRUE} \} \tag{6}

\[
x \in \text{IfcRelBuildingElement} \mid ((\text{A}(x) \text{ and B}(x) \text{ and C}(x)) \text{ or } D(x))
\]

A = \{ \text{SBRelBuildingElement}[\text{Exists}] = \text{TRUE} \} \tag{7}
B = { SB1IfcRelBuildingElement[Exists] = TRUE } (8)
C = { SB2IfcRelBuildingElement[Exists] = TRUE } (9)
D = { RelatedElement[Type] != 'IfcBuildingElementProxy' and (RelatedElement[Type] = 'IfcOpeningElement' and PhysicalOrVirtualBoundary[Value] = VIRTUAL) } (10)

For the second rule set (6) the original relation IfcRelSpaceBoundary (7) and the new relations IfcRelSpaceBoundary1stLevel (8) / IfcRelSpaceBoundary-2ndLevel (9) must be taken into account and only one of these alternatives may be used. At the same time, the usage of IfcBuildingElementProxy should be avoided and if an IfcOpeningElement is used the attribute PhysicalOrVirtualBoundary must be set to VIRTUAL (10).

The requirements described above facilitates the decision-making process in the validation for use of a building model in thermal simulation. The planned extensions to version 1.1 will make the definition of the rules easier and provide new capabilities, as previously described. Nevertheless, mvdXML still has limitations when it comes to checking geometric properties. Up to now, only processing of XML and IFC EXPRESS schema is necessary to perform mvdXML checks. However, as soon as geometric rules have to be taken into account, a geometry kernel is required.

**mvdXML Software Implementation**

Since the number of applications supporting mvdXML is limited, an openBIM validation module is implemented in the application FZKViewer (FZKViewer, 2020). It has also been shown that the existing tools are able to create valid mvdXML instance documents with respect to the mvdXML schema. However, these tools lack further checking mechanisms to validate the rules themselves and if the defined templates are valid according to the referenced IFC schema. At the same time, having its own implementation also enables the standard to be tested and further developed.

![Figure 6: openBIM workflow for checking building models](image)

The FZKViewer software tool is developed for the visualization, analysis and integration of different spatial data models. Focus is set on neutral semantic data models in the area of BIM and GIS such as IFC, CityGML and gbXML. Furthermore, this tool is used to perform and evaluate thermal building simulations based on the previously mentioned formats. Currently integrated are the simulation environments ETU-Simulation (Hottgenroth, 2021) and EnergyPlus (Crawley, 2021). This functionality is only available in the in-house application IFCExplorer.

The validation module enables an integrated openBIM workflow. If a model passes all checks successfully a thermal building simulation can be performed directly (see Figure 6). This means that building models from any openBIM compliant authoring software can be imported and analyzed. Errors that are identified can be exchanged with architects and engineers based on the BIM Collaboration Format (BCF), a buildingSMART standard developed since 2009. It allows BIM applications to exchange model-based issues. The communication of the issues can thus be realized file-based via BCF-XML (bSI BCF-XML, 2017) or by using the BCF-API (bSI BCF-API, 2017), a RESTful web service. Since support for the BCF-XML file format is currently more common than web service support, BCF-XML is also initially used in the workflow introduced in the present paper.

The implementation consists of three major components:

- **mvdXML Rule Engine**
- **mvdXML Editor / Validator**
- **mvdXML Report Generator**

The central component is the mvdXML Rule Engine which is responsible for the entire processing of the underlying IFC schema, the interpretation of the ConceptTemplate definitions and the TemplateRules. From the technical perspective, the entire implementation is realized in C++ and template metaprogramming (TMP) is used for the actual core of the mvdXML Rule Engine. This simplifies the use of different data types that have to be taken into account during the check. For the checks, the corresponding rules representing a hierarchical tree are connected by logical operators (and, or, xor, nand, nor, nxor and not). The values of the individual rules are checked using the typical comparison operators (==, !, >, >=, < and <=).

Another component represents the mvdXML Editor / Validator (Figure 7 bottom shows the user interface of the mvdXML Editor / Validator). This part supports reading and writing the current mvdXML versions 1.0 and 1.1 as well as the proposed extensions for an upcoming version. With the editor it is possible to create new mvdXML datasets and to modify existing ones. As far as possible, the user is assisted in entering and changing values to ensure the consistency and correctness of mvdXML datasets. This is especially relevant for UUIDs (Universally Unique Identifier) which are used for references within the mvdXML dataset, for example references from a Concept to an ExchangeRequirement or ConceptTemplate.

In addition, the transfer of rules to another IFC schema version is simplified. Corresponding changes are made automatically. An additional validation module analyses
whether the used ConceptTemplates correspond to the newly assigned IFC schema by checking the used entities and attributes.

The third component is the mvdXML Report Generator (see also Figure 7 top). After checking an IFC building model, an interactive report dialogue is presented. This dialogue provides a complete overview of the current test run. Sorted by Concepts, the dialogue displays whether a filter definition (Applicability) was applied and lists the entities tested successfully and those with error. A filter in the dialogue can be used to switch between a complete list of messages, errors messages and user-specific messages already mentioned. In particular, the presentation of user-specific messages is an important feature, as automatically generated messages derived from the IFC schema are usually not user-friendly. However, this only works if rules and messages are defined accordingly.

Within this dialogue the user can control BCF generation in detail. The corresponding BCF-XML file is then automatically created for exchange.

Conclusion

The present paper introduces a workflow for openBIM content quality check. This differ from the methodology proposed by buildingSMART, instead of following the process of creating an IDM, generating MVD and formalizing it using mvdXML, a more pragmatic way is applied. Issues, which occur in the exchange of architectural models with thermal simulation tools, are analyzed and corresponding rules are formulated. By collecting these rules, a comprehensive rule set is created successively. In this paper the identified checking rules are described briefly, and two rules are explained in detail. In order to test the current available rule set, the software FZKViewer is extended to perform model checks based on mvdXML. The implemented software modules are introduced and explained. Special focus is given to the reporting system including the possibility to exchange the checking messages via BCF.

Finally, a brief overview of a workflow for checking openBIM model in the context of thermal building simulation is given.

The software presented, as well as the checking rules developed within IBPSA Project1 can be downloaded under: https://www.iai.kit.edu/ifc

Acknowledgement

This work emerged from the IBPSA Project1, an international project conducted under the umbrella of the International Building Performance Simulation Association (IBPSA). Project1 will develop and demonstrate a BIM/GIS and Modelica Framework for building and community energy system design and operation.

References


Proceedings of the 17th IBPSA Conference
Bruges, Belgium, Sept. 1-3, 2021

ISO 29481 (2010). ISO 29481
ISO 16739
ISO 10303