IFC model checking based on mvdXML 1.1

M. Weise, N. Nisbet & T. Liebich
AEC3 Ltd., Germany, UK

C. Benghi
Northumbria University, Newcastle upon Tyne, UK

ABSTRACT: A significant barrier for successful use of BIM is the ability to efficiently and transparently agree on what data should be delivered by the many stakeholders of the supply chain and when. This requires additional agreements and specification work on top of existing standards like IFC. Ideally, these specifications are ready for automatic model checking to ensure the exchange of required BIM data. Based on the IDM/MVD methodology and the mvdXML 1.1 format developed by buildingSMART a web-based requirements management solution called BIM-Q and the mvdXML extension of the XBIM toolkit is discussed that demonstrates how BIM exchange requirements can be configured, managed and used for automatic model checking. All necessary steps are shown using an example from the STREAMER project, namely the Program of Requirements (PoR) and the early design of the room layout for hospitals. Besides presenting preliminary process implementation findings, grounded on data collected from various projects, persisting limitations for managing requirements and in particular for model checking based on mvdXML are discussed. An outlook of potential extensions and improvements of the different tools, mvdXML specification and the whole checking process is presented at the end.

1 INTRODUCTION

1.1 *Framework for BIM information management agreements*

BIM information management and control are fundamental processes in the delivery of the benefits enabled by the use of the BIM technologies. Currently, a significant barrier for beneficial use of BIM is the inability to efficiently and transparently agree on data exchange workflows across the many stakeholders of the supply chain. If expected data is missing, incorrect or misplaced then the project team has to go through additional data correction processes that can be time consuming and lead to critical project delays.

buildingSMART have developed guidance and standards that help defining a framework for BIM information management agreements; these are based on the production of Information Delivery Manuals (IDM) and the use Model View Definitions (MVD). An IDM is essentially an agreement on the processes and responsibilities of the project partners, whereas an MVD clarifies the data implementation details. Following the prescribed guidelines can result in time-consuming analysis, design and specification work that normally produces descriptive documents that later need to be implemented in software. The adoption of standard computer-interpretable formats, supported by efficient editing and management tools, would help to improve the requirement capturing and implementation processes.

1.2 *Solution approach and structure of the paper*

The work presented in this paper describes how the process of IDM/MVD development for IFC-based data exchange can be efficiently implemented with the use of the latest mvdXML 1.1 specification format through the adoption of a web-based requirements management tool called BIM-Q and the mvdXML extension of the XBIM toolkit.

Chapter 2 will introduce the workflow supported by the developed approach and will give an example defined in the STREAMER research project. All necessary steps are discussed and compared with the state-of-the art technology. The main focus will be on the following steps: (1) capturing data requirements as done by domain experts, (2) linking data requirements to processes, (3) specifying the mapping to IFC by configuring predefined concept templates, (4) the generation of a checkable mvdXML document and (5) the model checking in XBIM and the error reporting using the BIM collaboration format (BCF).

Chapter 3 will give an introduction to mvdXML release 1.1 being published as final version in 2016.

The focus will be on features that are relevant for the configuration of exchange requirements and automatic model checking. It will also clarify the scope of checking exchange requirements in order to avoid misunderstandings about the kind of quality checks that are in focus of the presented scenario and mvdXML.

Chapter 4 will present a solution to capture exchange requirements in a web-based environment, the BIM-Q tool. An important method for defining and managing requirements is the use of templates; available in the BIM-Q database as well as the mvdXML format, templates provide a key feature to reduce the complexity of the requirement definitions. Through templates, technical details can be embedded in preconfigured specifications files that, once refined by skilled specialists, simplify and modularize the usage and understanding of data requirements making them easily accessible by non IT-experts.

The following chapter presents the implementation of mvdXML model checking developed as a plugin for the xBIM Xploerer IFC viewer; it will present implementation objectives and details introducing options in the user interface that has been designed to support a goal oriented interaction with requirement specifications on the foundation provided by the structure of mvdXML.

Usage of the presented solution is shown in chapter 6 where examples from the STREAMER project are discussed to highlight different aspects of the solution provided along with an overview of its limitations. In addition to this, the conclusion in chapter 7 is discussing potential development directions.

2 EXAMPLE – SPACE REQUIREMENTS FOR HOSPITALS

2.1 *Introduction to the STREAMER case studies*

STREAMER is an industry-driven collaborative research project on Energy-efficient Buildings (EeB) that aims to reduce the energy use and carbon emission of new and retrofitted buildings in mixed-use healthcare districts. An important task in that scenario is to achieve unequivocal clarity about the client requirements.

In this particular case, for hospitals, this is achieved starting from the definition of space requirements, which need to be translated to space layouts following given design rules that take into account the constraints of the building site and existing buildings. In case of STREAMER the space layout is generated by an optimization algorithm, the Early Design Configurator, which produces a set of solutions that are evaluated against a set of KPIs, including energy consumption indicators. This simple workflow includes four processes and three data exchanges (see Figure 1).

In order to make sure that each process has a complete set of information the minimum exchange requirements are specified as an MVD using the mvdXML format. This enables to control IFC-based data exchange by checking existence of required information. This will ensure a certain level of quality.

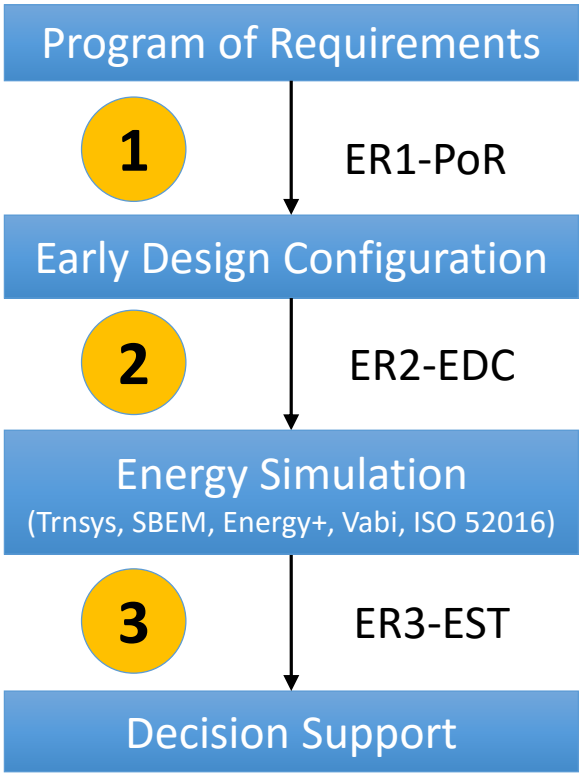

Figure 1 showing the workflow and exchange requirements comprising of (1) client requirement definitions, (2) the space layout, (3) energy simulation using various tools and (4) decision support.

2.2 *IDM/MVD methodology and state-of-the-art*

According to the Information Delivery Manual (IDM, ISO 29481-1) and Model View Definition (MVD) methodology the specification work follows subsequent steps and involves different stakeholders starting with a high-level view on the business processes down to software implementation details. The result of each step is a custom documented agreement or technical specification that forms the basis for further communication and refinements.

The work presented in this paper starts with the definition of Exchange Requirements. Accordingly, relevant processes, involved actors and the data flow as for instance presented in Figure 1 are already available as a reference. One out of the three mentioned data exchanges is the data defined by the Program of Requirements (PoR). Domain experts, in that case mainly the client, have to describe what information is captured in the PoR and ensure there's agreement on terms used, their meaning and the planned arrangement of required data. In case of PoR this results in a set of space types that are classified by criteria such as comfort, safety, hygiene

class, accessibility and others (Di Giulio 2015). For each of those classification criteria allowed ranges have to be defined including terms of parameter constraints, applicable design rules or required technical specifications (a space classified as "A4" means for instance that it should be accessible for staff only). Such classification systems are likely already available for the client and thus need only to be referenced.

The structure of defined requirements may partially fit to other processes. Therefore, it is reasonable to harmonize specifications by reusing them in other processes. If the room type information is needed for space layout but also for energy estimation it should be linked as a requirement to both processes. Traditionally, the main purpose of this step is to prepare implementation of software interfaces, which means to translate the terms of domain experts to a data structure like IFC. This step is done by modelling experts who are familiar with the relevant data structure. For IFC-based MVD developments it means to switch to the ifcDoc tool that enables to work on an mvdXML specification, but will lose the link to the exchange requirements defined by the domain expert.

Today, an MVD even if available as mvdXML is defined mainly for documentation purposes. The ifcDoc tool for instance enables to generate the HTML documentation as known from the IFC4 specification. This is expected to change if mvdXML-based model checking becomes available as presented in this paper. If such MVD specification enables to validate an IFC dataset, then it would not only support software implementation but also the everyday data quality control in real projects. If some requirements are not met the sender could be notified and pointed to the missing data. This can be done via the BIM collaboration format (BCF), which enables to report and visualize identified issues. Ideally, issues are reported using the terms from the original requirements definition and not the attribute or class name of IFC. This would improve communication.

The next chapter will highlight the checking and configuration features of mvdXML, while chapter 4 will detail how it can be generated by our requirements management environment to supports process-specific configurations.

3 MVDXML 1.1

3.1 *Overview and main use cases*

After a two year review period the mvdXML 1.1 specification was published in spring 2016 (Chipman et al.). Besides a couple of minor improvements and simplifications the most notable change is the extended capability for model checking. Although this feature of mvdXML received bigger attention lately the focus is still on MVD documentation purposes, for instance for creating the HTML documentation of the new Design Transfer and Reference View of IFC4. It might also be used for generating an IFC subset schema or data filtering, but both scenarios seem to be less important at the moment.

Although each of those usages has specific requirements the definition of an MVD is always similar. Main elements of each MVD are:

- *ModelView:* one or more of those elements are normally included in an mvdXML file. It is part of the *View* element and is the main container for exchange requirements and root concepts.
- *ExchangeRequirement*: represents the data that is relevant for a use case, either for import, export or both.
- *ConceptRoot:* represents a class of objects for which the same constraints apply. They are normally linked to entities that are derived from IfcRoot, i.e. being a main testable element of an IFC model.
- *Concept:* is part of a root concept and defines a constraint on applicable objects and how it is used in exchange requirements.
- *ConceptTemplate*: defines a unit of functionality that is used and configured by ConceptRoot and Concept elements. It is a selection and basic configuration of IFC definitions that are required to implement a specific functionality such as support of property sets, material layer definition or more complex data like brep geometry.

Each of those elements is able to carry additional meta-data and descriptive text including multilingual support.

3.2 *Concept templates and their configuration*

An important feature in terms of reducing the maintenance effort is the use of configurable concept templates. A concept template defines one or more applicable entities and includes a set of rules that each specifies a sub graph of instantiable attributes. Such sub graph is defined by attribute and entity rules and always starts with an attribute of the applicable entity.

The concept template shown in Figure 2 is defined for all instances of *IfcRoot* entities and contains two rules for the attributes *Name* and *Description*. Both rules define an additional (optional) rule identifier (*RuleID*), which is a unique name used for further configuration. The figure also shows that an *AttributeRule* is followed by (one or more) *EntityRule* that expand the sub graph.

The rule identifier is later used as a parameter in a logical expression to check existence, values, types, size of sets or uniqueness. Accordingly, above shown example enables to configure both attributes, for instance to check for a specific name or existence of a description. However, logical expressions in

mvdXML are limited in their expressiveness in order to be as clear as possible both for definition and processing.

```
<ConceptTemplate
 uuid="c19ec186-9cfd-47fc-a4d4-9fb35008d04a"
 name="User Identity" applicableSchema="IFC4"
 applicableEntity="IfcRoot">
  <Definitions><Definition>
    <Body><![CDATA[Code 020- ...]]></Body>
  </Definition> </Definitions>
  <Rules>
  <AttributeRule RuleID="Name"
                 AttributeName="Name">
    <EntityRules>
      <EntityRule EntityName="IfcLabel"/>
    </EntityRules>
  </AttributeRule>
  <AttributeRule RuleID="Description"
                 AttributeName="Description">
    <EntityRules>
      <EntityRule EntityName="IfcText" />
    </EntityRules>
  </AttributeRule>
  </Rules>
</ConceptTemplate>
```

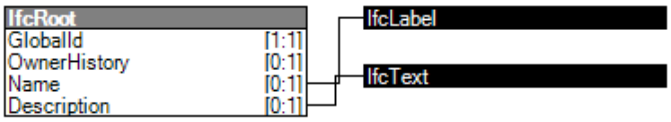

Figure 2 showing the ConceptTemplate "User Identity" and its visual representation as instantiation diagram.

3.3 *Checking of exchange requirements*

The principle for defining constraints is based on IF THEN statements. The IF-part is defined in *ConceptRoot* nodes and determines the selection of instances in the IFC model. The THEN-part is defined by *Concept* nodes and defines the constraints that shall be applied to all selected instances. In addition to a "selection by type" (through the *applicableEntity* field) it is possible to define additional constraints. For instance if all load bearing walls shall be checked then all instances of *IfcWall* with a property *Pset_WallCommon.Load-Bearing = TRUE* must be selected. Such additional constraints are defined in the <*Applicability*> section of *ConceptRoot*. The mvdXML snippet shown in Figure 3 is selecting instances of IfcBeam with the Name "Beam-206". It is configuring the concept template of Figure 2.

The configuration of constraints works in a similar way; a concept refers to a concept template using its uuid. The <*Requirements*> section then defines the link to exchange requirements and its expected usage. The configuration of rule identifiers starts thereafter, which may be using nested statements logically combined by Boolean operators. Figure 4 shows the configuration of a mandatory space property where only the two values "A1" and "A3" are allowed.

```
<ConceptRoot
 uuid="00000035-0000-0000-2000-000000067001"
 name=" Beam-206"
 applicableRootEntity="IfcBeam">
  <Applicability><Template
  ref="c19ec186-9cfd-47fc-a4d4-9fb35008d04a"/>
  <TemplateRules operator="and">
    <TemplateRule
      Parameters="Name[Value]='Beam-206'"/>
  </TemplateRules>
  </Applicability>
```

Figure 3 showing the configuration of "User Identity" for the selection of an IfcBeam instance.

```
<Concept
 uuid="00000003-0000-0000-0000-000000349910"
 name="Accessibility Labels">
  <Template
   ref="00000000-0000-0000-0001-000000000001"/>
  <Requirements>
  <Requirement applicability="import"
   exchangeRequirement="00000003-0000-0000-
   0000-000000000105" requirement="mandatory"/>
  </Requirements>
  <TemplateRules operator="and">
    <TemplateRules operator="or">
      <TemplateRule Parameters=
      "Set[Value]='STREAMER_Labels_PoR' AND
      Property[Value]='AccessSecurity' AND
      Value[Value]='A1'"/>
      <TemplateRule Parameters=
      "Set[Value]='STREAMER_Labels_PoR' AND
      Property[Value]='AccessSecurity' AND
      Value[Value]='A2'"/>
    </TemplateRules>
  </TemplateRules>
</Concept>
```

Figure 4 showing the constraint for the "Accessibility Labels" defined by the PoR for spaces.

4 CAPTURING REQUIREMENTS WITH BIM-Q

4.1 *Need for a shared, web-enabled requirements management tool*

As outlined in chapter 2.2 exchange requirements are a means for communication and thus need to be agreed and shared between involved participants. Also, many requirements are applicable for several processes so that a lot of definitions can and should be reused.

Today, exchange requirements are typically captured in a spreadsheet format. For each physical or conceptual thing it captures relevant properties, its

meaning and use in design processes (IDM). It is simple and straight forward but the more information is captured and shared, the more difficult it is to keep consistency and maintain the content. There are also limitations to evaluate and export requirements, in particular for generating various reports and producing an mvdXML file for checking purposes. Accordingly, there is a need for better tool support which was leading to the development of the presented web-based solution called BIM-Q.

4.2 *Scope related to the IDM/MVD methodology*

Before collecting exchange requirements an initial set-up of the database is necessary. The first step is to define a template guideline that shall group all definitions. This might later be used to configure project requirements. Next to this, the selection of involved stakeholders, covered stages and processes as well as relevant mappings is necessary. Mappings include links to classification systems, translations to other languages and the representation in data structures like IFC. In this initial step it means to set-up the boundaries for the discussed use cases in terms of definitions and standards that becomes relevant to clarify the meaning of terms and to be used for data exchange. Each of those settings can be changed or extended in later stages, but it defines the starting point for defining relevant terms, which is the first main step of capturing domain knowledge.

4.3 *Set-up of reusable concepts*

Definition of exchange requirements follows the object-oriented modelling principle, but with less restrictive rules. Everything is a concept. Each concept can be described, typed, mapped to other definitions and arranged to each other in order to form more complex concept definitions. A concept can for instance represent a class of beam objects whereas another concept represents a simple datatype property for fire rating.

An exchange requirement is typically defined for a property of some object class. A fire safety calculation may requires the fire rating property for all loadbearing building elements. It is a simple and natural way of expressing requirements that can be defined by non-IT experts.

Experiences have shown that a lot of concepts are reused for requirement definitions, in particular in case of generic properties. This is leading to a lot of copied content that is later difficult to maintain. Therefore, the first step is to collect reusable concept definitions that can be arranged in any level of complexity. In that way, a pool of concepts is defined that later can be arranged to any requirement setting that needs to be described. Each reusable concept is linked to default definitions, such as a description or the mapping to IFC, which however can be overridden in a requirement setting if necessary.

The pool of reusable concepts can be organized according to own preferences. Our recommendation based on experiences is to organize similar concepts in groups like classes, properties and geometry. STREAMER is using a labelling approach and thus is using the structure as shown in Figure 5. Further subgroups are recommended, but should be kept as simple as possible. If properly arranged it later helps to find the right concept and to configure the requirement settings.

Figure 5 Reusable concepts as defined in the STREAMER project.

4.4 *Configuration of exchange requirements*

The next step is to link objects with properties in order to express requirements. This is done by dragging reusable concepts to a new requirements tree as shown in Figure 6. Both trees provide independent search capabilities so that concepts can easily be found and arranged in the requirements tree. In order to speed-up the set-up process it is also possible to drag and drop a concept with all child elements. If reusable concepts are properly arranged it supports an easy and fast set-up process.

Differently to reusable concepts there are some constraints regarding the organization of the requirements tree. Those constraints exist mainly due to the fact that some meaningful reports or an mvdXML file shall be generated out of this tree. By following the idea of having a property of some object class the structure should follow the rule of having a property concept, marked as a simple datatype, always as a child element of an object concept. In between there might additional group concepts for better organization of requirements, which are ignored for later model checking. There are special so-

lutions for enumeration datatypes having allowed values as child concepts, which however do not break described general rule. Nevertheless, a risk of configuring a requirements tree that cannot be properly exported to mvdXML checking file remains so that this step should carefully done.

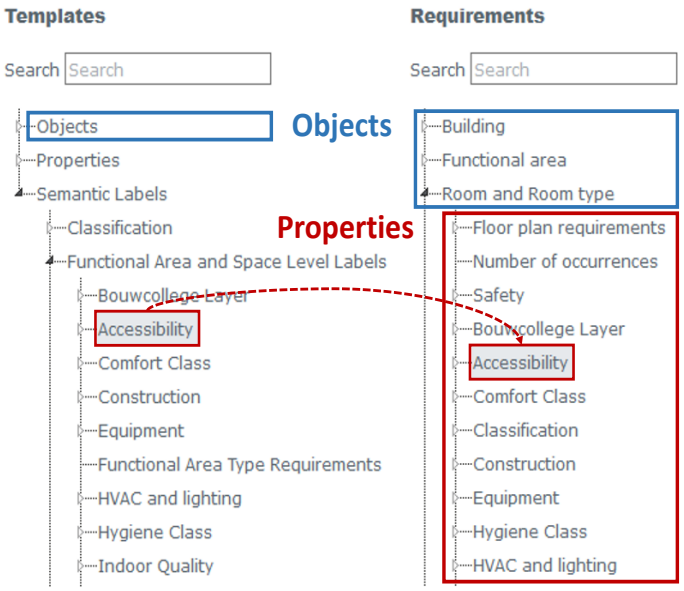

Figure 6 Set-up of the requirements tree by dragging reusable concept from templates (left) to the requirements tree.

Once the requirements tree is defined the usage settings for the different processes can be configured. It basically means to make a decision what data is required, optional or not allowed. Additionally, an owner of a data concept has to be defined who is responsible to deliver that information (Figure 7).

4.5 *IFC mapping definitions*

Each concept can have any number of mapping definitions to whatever data structure is of interest. In our case the focus is on the open IFC-BIM format that can be formalized by mvdXML definitions.

There are basically two types of mapping definitions:

– *Object concept mappings:* For mvdXML it means to configure a *ConceptRoot* element comprising of the selection of an IFC entity (*applicableRootEntity*) and, optionally, additional *Applicability* settings.
– *Property concept mappings:* This requires the configuration of a *Concept* element, which needs to identify and configure an appropriate *ConceptTemplate*.

The BIM-Q tool supports a simple syntax to easily configure most frequently needed mapping definitions. An object concept for instance maps either 1:1 to an IFC entity, or is additionally restricted by the *PredefinedType* attribute or some property values. The expression *IfcWall.IfcWallTypeEnum.SHEAR* is for instance applicable for all *IfcWall* instances having the *PredefinedType* attribute set to "*Shear*". Similar solutions are available for property concepts, where for instance the configuration of properties and quantities is often needed. Uncommon mapping definitions have to go through a more complex configuration process. This however shouldn't be a problem as this step has to be done by an IFC expert who is familiar with the IFC specification and available mvdXML concept templates.

Concept Definition	Owner	ER1-PoR	ER2-EDC
Building	Architect	-	MAN
Functional area	-	MAN	MAN
Room and Room type	Architect	MAN	MAN
Accessibility	Building owner	MAN	-
Bouwcollege Layer	Building owner	MAN	-
Classification	Architect	MAN	MAN
Comfort Class	Building owner	MAN	-
Construction	Building owner	MAN	MAN
Containment	Architect	-	MAN
Equipment	Building owner	MAN	MAN
Geometric representation	Architect	-	MAN
HVAC and lighting	Building owner	OPT	-
Hygiene Class	Building owner	MAN	MAN
Indoor Quality	Building owner	OPT	-
Number of occurrences	Building owner	MAN	NOT

Figure 7 Definition of usage settings and assignment to a concept owner

4.6 *Reporting and mvdXML export*

The final step in the requirements capture process is to produce some sort of evaluable result. This might be a specific PDF report that could act as an contract annex, an mvdXML file for checking purposes or some template documents. In case of mvdXML it is possible to export all settings to a single file. Alternatively, it is also possible to export settings of specific processes or a single owner only.

The export feature itself is translating the used mapping syntax to an mvdXML, which for instance in case of properties expands to a check of *properties on occurrences* and *properties on types*. At the time of this writing there is no consistency check against the IFC specification so that spelling errors are not identified. However, testing a valid file should quickly show wrong mapping definitions.

5 MODEL CHECKING WITH XBIM

5.1 *mvdXML implementation*

In order to test the adoption of mvdXML-based requirement specifications against the model data exchanged between different stakeholders of the STREAMER project, an implementation of the validation features of mvdXML 1.1 has been developed using the infrastructure offered by the open source xBIM toolkit.

The implementation is mainly designed to allow individual stakeholders to independently verify the conformity of received and produced IFC models against the agreed exchange requirements and concept roots in a user friendly visual 3D environment.

To maximize the reusability of the developed components in other validation scenarios the implementation has been divided into two software components:

1. the mvdXML validation library (mvdLib) is a .NET dynamic link library providing validation capabilities that can be consumed in multiple deployment scenarios (e.g. Xplorer UI, web services, cloud environments, command line applications, etc.)
2. the XbimXplorer mvdXML Plugin (mvdUi) is an extension plugin for the pre-existing XbimXplorer IFC viewer that provides the User Interface for interactive validation of models against specification files.

Both modules have further development activities planned in response to feedback from the STREAMER project as well as from scheduled innovations in the underlying xBIM toolkit.

5.2 *User interface development and collaboration workflow*

To enable a complete collaboration workflow between stakeholders of the established IDM processes the mvdUI component has been designed to allow the interactive analysis of models according to arbitrary combinations of exchange requirements, concept roots and IFC classes, the UI allows immediate feedback on the validation status of selected elements as well as whole models; this filtering strategy also helps to improve the responsiveness of the application which can become relevant if thousands of requirements need to be checked for large IFC models. Visual color coding styles have been developed to allow rapid traffic-light model inspection in the 3D viewer of passing and failing requirements.

The development of features for the semi-automatic production of validation reports in the BIM collaboration format (BCF) have required the redesign of the XbimXplorer plugin API in order to allow integration of the MVD plugin with the existing BCF plugin; the designed features allow stakeholders to exchange communication threads on the result of validation tests across different BIM platforms while retaining complete reference of the involved IDM, MVD and IFC background.

6 PROOF OF CONCEPT

6.1 *Preparing client requirements (ER1-PoR)*

Much of the client requirements is shared through informal spreadsheets. In the current case the PoR was prepared using BriefBuilder and the information was shared as a simple CSV file. In order to make this information available for formal checking prior to incorporation into the design process, it is necessary to add the semantic meaning of the individual rows and columns. This was achieved through the use of the AEC3 BimServices Transform1 utility.

The semantic meaning of the rows is by default unknown. The transformation takes a single extra parameter ‘topic’ which identifies the semantic object represented by the rows. The choices include ‘project’, ‘site’, ‘building’, ‘storey’, ‘zone, ‘component’, ‘system’, ‘type’ or in this case ‘space’. The transformation then creates a complete IFC model with the minimum number of other objects necessary to give context for the objects.

Each data field is mapped to a property grouped in a default property set ‘*Default_SpaceProperties*’. However, the transformation makes use of a global dictionary which contains hints which can add value to the outcome by associating the column headers to specific IFC attributes (Figure 8). The global dictionary can also hold pointers to the expected parent, for example a property set, any synonyms, and any expected values.

```
<concept type="property">
  <term context="BriefBuilder">Room type</term>
  <term context="PoR">RoomType</term>
  <term context="IFC">ObjectType</term>
  <term context="en-GB">
    Space or Component Type</term>
</concept>
```

Figure 8 Example from the global dictionary to control the CSV to IFC mapping.

6.2 *Checking requirements*

Checking of the resulting IFC model in XBIM is straight forward and shown in Figure 9. After importing both the IFC and the mvdXML file the exchange requirement can be selected and checked (right top view). All constraints that passes or fails are shown with traffic lights in the window below. Each test result is linked with the object in question and can be browsed in the 3d viewer and the properties window. For supporting the communication within the design team a BCF issue can also be generated to point to failures.

Various filter options enable to focus on specific objects or constraints. It is for instance possible to select specific object types or properties of an exchange requirement only. It is also possible to select elements in the 3D view that shall be checked by selected requirement definitions. This feature also helps to improve performance in case of very big IFC files and/or constraints.

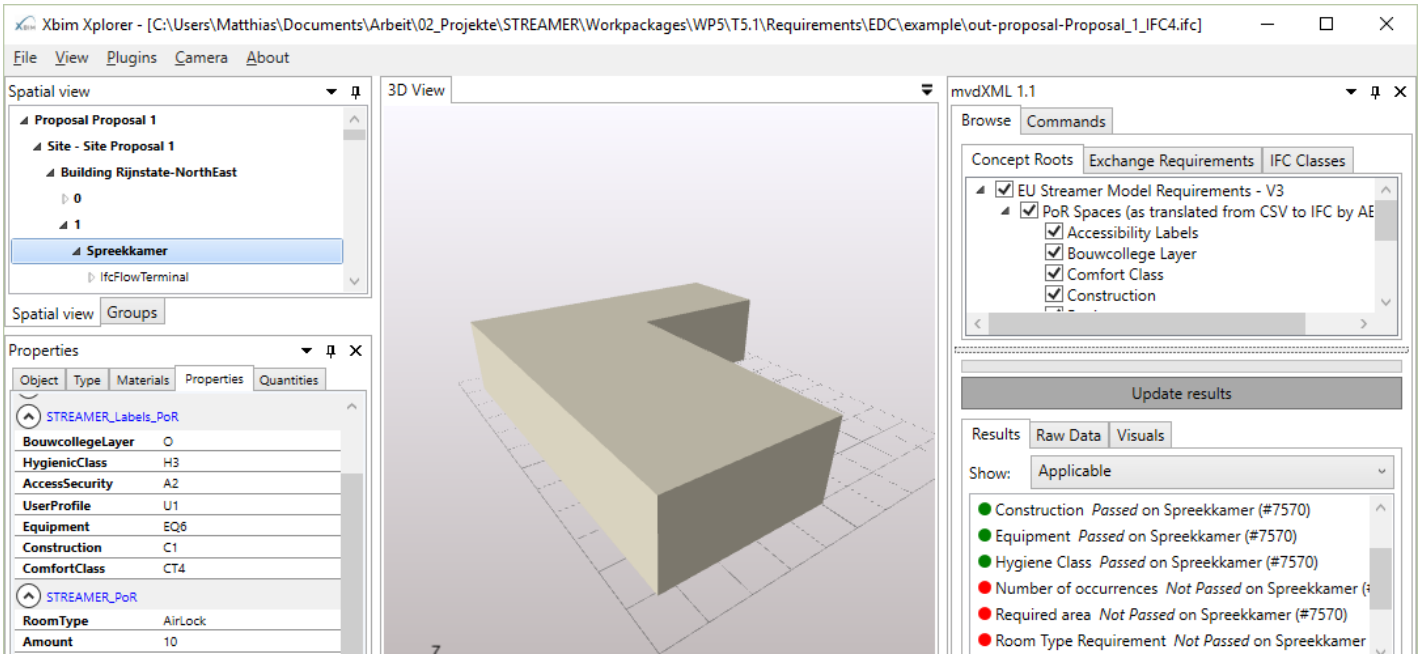

Figure 9 Checking result of an example space layout generated from space requirements.

7 CONCLUSION AND OUTLOOK

An integrated approach for checking exchange requirements based on the mvdXML 1.1 format has been presented. Requirements management is an essential element and is supported by a novel web-based solution called BIM-Q, which not only enables to capture, maintain and easily configure exchange requirements but also allows to specify its mapping to IFC and the generation of an mvdXML file. The mvdXML-based model checking was implemented as a plugin for the open source XBIM viewer and has been validated with examples from the STREAMER project.

Current implementation shows the overall potential of that development. It enables to improve the quality of BIM-based data exchange. However, further research is necessary to develop best practices and more templates in order to reduce the specification effort. Instead of starting from scratch the user can then reuse available requirement definitions and can focus on project specific configurations. Another field of development is to provide consistency checks of configured requirements. And last not least it has to discussed if and how to extend the checking capabilities of mvdXML in order to go beyond yet available fundamental checks.

ACKNOWLEDGMENT

The presented research was done in the frames of the European FP7 Project STREAMER and the buildingSMART Norway BIM Guide developments. We acknowledge the kind support of the European Commission, bS Norway and the project partners.

REFERENCES

BCF.
http://www.buildingsmart-tech.org/specifications/bcf-releases

BriefBuilder.
http://www.briefbuilder.nl/

Chipman, T. Liebich, T. & Weise, M. 2016. mvdXML – Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules. *buildingSMART International Ltd. 15.02.2016.*

Di Giulio, R. Quentin, C., van Nederpelt, S. Traversari, R. Nauta, J. & Turillazzi B. 2015. D1.2: Semantic typology model of existing buildings and districts. *Deliverable of the STREAMER project.*

IDM.
http://iug.buildingsmart.org/idms/

IFC4 Design Transfer View.
HTML documentation: http://www.buildingsmart-tech.org/mvd/IFC4Add1/DTV/1.0/html/

IFC4 Reference View.
HTML documentation: http://www.buildingsmart-tech.org/mvd/IFC4Add1/RV/1.0/html/

ifcDoc tool.
http://www.buildingsmart-tech.org/specifications/specification-tools/ifcdoc-tool/ifcdoc-beta-summary

ISO 29481-1:2010. Building information modelling - Information delivery manual - Part 1: Methodology and format. *Published by ISO/TC 59/SC 13.*

MVD.
http://www.buildingsmart-tech.org/specifications/mvd-overview/mvdxml-releases/mvdxml-1.1

Nisbet, N. 2010. BimServices – Command-line and Interface utilities for BIM. http://www.aec3.com/en/6/6_04.htm

STREAMER website.
http://www.streamer-project.eu/

XBIM websites.
http://www.openbim.org/, https://github.com/xBimTeam