mvdXML

Requirements and Examples

Review of a standardized format to define and exchange   
Model View Definitions with Exchange Requirements and Validation Rules

by  
mvdXML Group of buildingSMART International Ltd.  
(with kind support from:   
Tim Chipman, Thomas Liebich, Matthias Weise, Jakob Beetz, Léon van Berlo, Chi Zhang,   
Donghoon Yang, Chuck Eastman, ..)

Preface

This document collects requirements and examples for mvdXML. It is a working document developed in the review process that extends the current draft proposal of mvdXML 1.1. The main goal is to clarify the use of mvdXML, to identify existing gaps and to guide further developments. It is likely that examples developed in this document will be integrated in the final release of the mvdXML documentation.

The document contains five chapters:

1. Requirements, which collects all requirements for mvdXML 1.1 and upcoming versions. It tries to categories rule types and to be as short as possible.
2. Examples, which describes reals use cases taken for instance from experiences with IFC implementation and ongoing MVD developments.
3. Agreements, which summarizes additional decisions for the use of mvdXML.
4. Discussions, which is a collection of questions and answers within the mvdXML group.
5. Tools and Reference Implementations, which gives an overview about available tools.

Table of Content

[1 Requirements 5](#_Toc374096642)

[1.1 Data validation rule types 5](#_Toc374096643)

[1.1.1 Existence of attributes and references 5](#_Toc374096644)

[1.1.2 Existence of elements (global and local) 5](#_Toc374096645)

[1.1.3 Content of simple data type values 6](#_Toc374096646)

[1.1.4 Size of aggregates 6](#_Toc374096647)

[1.1.5 Content of aggregates 6](#_Toc374096648)

[1.1.6 Uniqueness (global) 7](#_Toc374096649)

[1.1.7 Uniqueness (local) 7](#_Toc374096650)

[1.1.8 Type of referenced data 7](#_Toc374096651)

[1.1.9 Conditions 8](#_Toc374096652)

[1.1.10 Nesting and recursion 8](#_Toc374096653)

[1.1.11 Checks based on mathematical operations 8](#_Toc374096654)

[1.1.12 <TBE> 9](#_Toc374096655)

[1.2 MVD Documentation 9](#_Toc374096656)

[1.3 Specification of subset schemas 9](#_Toc374096657)

[1.4 Data filtering 9](#_Toc374096658)

[2 Examples 10](#_Toc374096659)

[2.1 Implementer Agreements for IFC 2x3 (adopted to IFC4) 10](#_Toc374096660)

[2.1.1 #CV-2x3-100: correct usage of IfcMaterialLayerSetUsage for IfcSlab 10](#_Toc374096661)

[2.2 “Model Checking” use-cases from the Dutch AEC industry 11](#_Toc374096662)

[*2.2.1* *Use case 1: Psets* 11](#_Toc374096663)

[*2.2.2* *Use case 2: Masonry supplier* 11](#_Toc374096664)

[*2.2.3* *Use case 3: Fire resistant doors* 11](#_Toc374096665)

[*2.2.4* *Use case 4: Is there a door?* 11](#_Toc374096666)

[*2.2.5* *Use case 5: Is an object on the storey it should be?* 11](#_Toc374096667)

[*2.2.6* *Use case 6: Elevation* 11](#_Toc374096668)

[*2.2.7* *Use case 7: Material* 11](#_Toc374096669)

[*2.2.8* *Use case 8: classification consistency* 11](#_Toc374096670)

[*2.2.9* *Use case 9: program requirements* 11](#_Toc374096671)

[*2.2.10* *Use case 10: Breps versus extrusions* 12](#_Toc374096672)

[2.3 Validation testing for the Precast BIM Standard 13](#_Toc374096673)

[3 Agreements 14](#_Toc374096674)

[3.1 Exclude subtypes from referenced entities 14](#_Toc374096675)

[3.2 Using unnamed INVERSE relationships 14](#_Toc374096676)

[4 Discussions 15](#_Toc374096677)

[4.1 Configuration of min and max cardinality 15](#_Toc374096678)

[5 Tools and Reference Implementations 15](#_Toc374096679)

[5.1 IFC Documentation Generator 15](#_Toc374096680)

[5.2 Constructivity Viewer 15](#_Toc374096681)

[5.3 GTDS 15](#_Toc374096682)

[5.4 BIMServer 15](#_Toc374096683)

[6 Appendix 16](#_Toc374096684)

# Requirements

mvdXML is developed to support 4 main use cases, namely (1) MVD documentation, (2) specification of subset schemas, (3) data filtering and (4) data validation (see draft proposal for mvdXML 1.1). The current focus of mvdXML 1.x is set to use cases (1) and (2). Use cases (3) and (4) are getting more and more import to support BIM-based work, and since most questions and requirement definitions relate to these use cases, especially data validation, they are in focus of this working document.

## Data validation rule types

The following section is a proposal to categorize rule types for data validation and sets a link to further examples. As basis for more detailed discussion the matrix developed by Donghoon Yang is used (see Appendix).

### Existence of attributes and references

|  |  |
| --- | --- |
| **Requirement** | Specific attributes or references of an instance should be present or should not be present. |
| **Examples** | 1. IfcSpace.Name shall exist 2. …   See also:   * Use case 1: Psets |
| **Review** | Supported |
| **Recommendation** | No change or extension needed (?) |

### Existence of elements (global and local)

|  |  |
| --- | --- |
| **Requirement** | An instance of a specific entity type shall be available. |
| **Examples** | An IFC model should contain   1. one instance of IfcProject and 2. at least one instance of IfcBuilding (being a part of the spatial structure).   See also:   * .. |
| **Review** | Not supported  mvdXML 1.1 does not support to require instances of a particular entity type. Therefore, it is not possible to deal with the first example. However, this particular requirement (having always one instance of IfcProject) is a major constraint for any IFC model and therefore is already defined in the IFC Express schema.  The second example could be managed in a different way because an IfcProject instance is the root element of an IFC model and enables to access all contained data through references. Therefore, this constraint could be defined by checking references of IfcProject instances. |
| **Recommendation** | 1. make sure that all entities can be reached by references from the root entity (may enable to use “unnamed” inverse references) -> may add implementer agreement (see 3.2) |

### Content of simple data type values

|  |  |
| --- | --- |
| **Requirement** | An attribute should have a specific value or should be within a given range.  This requirement is related to simple data types, including enumeration values. |
| **Examples** | 1. A string value must be equal to “Test” 2. A integer value must be greater than 0 3. A string value shall contain the substring “-Room” 4. .. |
| **Review** | Supported |
| **Recommendation** | No change or extension needed (?) |

### Size of aggregates

|  |  |
| --- | --- |
| **Requirement** | An aggregate (LIST, SET, BAG, ARRAY) shall be limited by min and max cardinality settings. |
| **Examples** | <TBD>  Links:  See discussion about Configuration of min and max cardinality |
| **Review** | Supported by   1. EntityRule.Cardinality and AttributeRule.Cardinality on ConceptTemplate level 2. Metric [Size] on ConceptTemplate and Concept level |
| **Recommendation** | Clarify how to configure a AttributeRule through metric [Size] +  enable to set a link from TemplateRule to AttributeRule (otherwise aggregations attached to the root concept cannot be constrained by metric [Size]).  See also: Metric [Size] for AttributRule |

### Content of aggregates

|  |  |
| --- | --- |
| **Requirement** | Existence (or absence) of specific elements/values shall be defined.  Additionally, for LIST and ARRAY a sequence of elements or values, maybe including a fixed position, shall be defined. |
| **Examples** |  |
| **Review** |  |
| **Recommendation** | Implementer agreement needed. |

### Uniqueness (global)

|  |  |
| --- | --- |
| **Requirement** | A value of an attribute shall be unique for all instances. |
| **Examples** | 1. ID or naming of an element shall exist only once within a data model. |
| **Review** | mvdXML grammar supports the metric [Unique], which does not differentiate between global and local uniqueness. |
| **Recommendation** | Agreement needed how to differentiate between global and local uniqueness. The following is suggested:   * Simple data types or single references -> global uniqueness * Aggregation data types -> local uniqueness (no same value within the aggregation)   Open issue with this proposal:  Global uniqueness for aggregation data types cannot be defined. |

### Uniqueness (local)

|  |  |
| --- | --- |
| **Requirement** | A value within an aggregation of values shall be unique. (only needed for BAG, LIST and ARRAY) |
| **Examples** |  |
| **Review** | See: Uniqueness (global) |
| **Recommendation** | See: Uniqueness (global) |

### Type of referenced data

|  |  |
| --- | --- |
| **Requirement** | The entity type of allowed references shall be restricted. |
| **Examples** | The following restriction are possible   1. a subset of select data types 2. a subset of the subtypes of referenced entity type. 3. the referenced entity type only 4. all subtypes including the referenced entity type, i.e. no restriction. |
| **Review** | All of above requirements are supported. The following remarks are made:   * Type 1: all selectable data types must be added to *AttributeRule.EntityRules* * Type 2: all usable subtypes must be added to  *EntityRule.EntityRules* * Type 3: requires to add a self-reference to *EntityRule.EntityRules* (otherwise all subtypes are valid – see type 4) * Type 4: no additional definitions are necessary (all subtypes are included by default)   NOTE: Using the default setting can reduce definition efforts. But it means that nothing can be specified about the use of subtypes and their specific attributes. Accordingly, it is only useful if referenced entity types are defined somewhere else, e.g. in a ConceptRoot. |
| **Recommendation** | Improve documentation,  Agreement for Type 3 (see 3.1) |

### Conditions

|  |  |
| --- | --- |
| **Requirement** | A constraint shall apply only if a condition is fulfilled. |
| **Examples** |  |
| **Review** | Supported, but may needs simplification (?)  mvdXML 1.0 introduces conditional statements by using inner and outer parameters. An outer parameter describes the condition and the inner parameter defines the validation constraint. Since mvdXML 1.1 it is also possible to use OR logic and grouping of constraints. A conditional statement:   IF Condition1 THEN Constraint1  could be defined like this:   (Condition1 AND Constraint1) OR (NOT Condition1).  NOTE: It is not possible to use parameters from different Concept Templates. In that case, a new Concept Template must be defined that contains all required parameters. |
| **Recommendation** | May add if-then statement to the grammar to simplify definition of conditions. |

### Nesting and recursion

|  |  |
| --- | --- |
| **Requirement** | Hierarchy depth of elements shall be restricted. |
| **Examples** | Element assemblies shall be restricted to   1. Max depth = 1 2. Max depth = n |
| **Review** | No recursion supported – only possible if “stateless” conditions can be specified to restrict hierarch depth (e.g. example 1: element is either a part or an assembly, but not both) |
| **Recommendation** | Out of scope for mvdXML 1.x  Postponed for later versions (check if existing solutions can be integrated) |

### Checks based on mathematical operations

|  |  |
| --- | --- |
| **Requirement** | Check if a value that must be derived from other values by using mathematical operations fulfils a constraint. |
| **Examples** | The area of a room must be greater than # m², and area needs to be calculated. |
| **Review** | Not supported |
| **Recommendation** | Out of scope for mvdXML 1.x Postponed for later versions (check if existing solutions can be integrated) |

### <TBE>

|  |  |
| --- | --- |
| **Requirement** |  |
| **Examples** |  |
| **Review** |  |
| **Recommendation** |  |

## MVD Documentation

Requirement related to documentation use case. <TBD if of interest>

## Specification of subset schemas

Requirement related to generation of subset schemas. <TBD if of interest>

## Data filtering

Requirement related to data filtering use case. <TBD if of interest>

# Examples

## Implementer Agreements for IFC 2x3 (adopted to IFC4)

This chapter contains selected implementer agreements for further discussion. Note that not all parts of an implementer agreement might be of interest or can be represented in mvdXML. Supported parts are explicitly mentioned in the table.

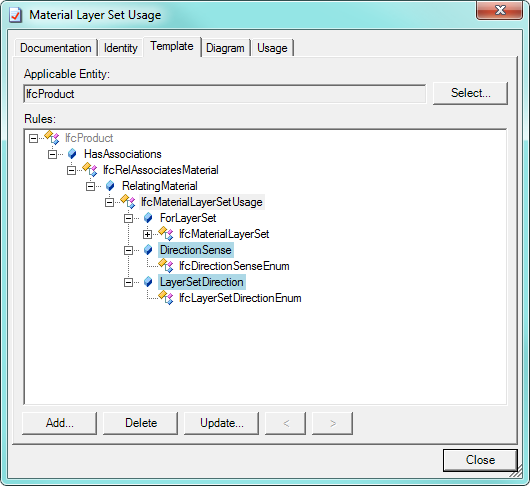
### #CV-2x3-100: correct usage of IfcMaterialLayerSetUsage for IfcSlab

|  |  |
| --- | --- |
| Root | IfcSlabStandardCase (new in IFC4) |
| Supported | Existence of attributes and references   * *IfcMaterialLayerSetUsage is mandatory*   Content of simple data type values   * *IfcMaterialLayerSetUsage.DirectionSense = Positive* * *IfcMaterialLayerSetUsage.LayerSetDirection = AXIS3* |
| Not supported | Checks based on mathematical operations + Conditions   * *The TotalThickness of the IfcMaterialLayerSet is the sum of all layer thicknesses and in case of …*   Other consistency checks |
| Link | <http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-100> |

**Example-CV100.xml**

Concept Template: The example defines a concept template *“Material Layer Set Usage”* that can be used by IFC entities defining a planar building element with constant thickness such as *IfcWallStandardCase* or *IfcSlabStandardCase*. It defines all data that is needed to define material layers. For configuration purposes it adds two parameters: (1) *DirectionSense* and (2) *LayerSetDirection*. Both parameters are used to define constraints for values.

NOTE: Data that is mandatory and sufficiently defined by the underlying schema might be omitted in the concept template definition. However, for documentation purposes it is recommended to include all required data as it improves clarity and comprehensibility of a concept template.



Concept: One concept of above shown concept template is defined for *IfcSlabStandardCase*. It is defined to be mandatory for the exchange requirement “Examples”. It also adds a template rule that defines allowed values for *DirectionSense* and *LayerSetDirection* using the following expression string:

DirectionSense[Value]='POSITIVE' AND LayerSetDirection[Value]='AXIS3'.

### #CV-2x3-104: number of doors and windows within one opening is restricted to max 1.

|  |  |
| --- | --- |
| Root | IfcOpeningElement and IfcOpeningStandardCase (for global check of all openings) IfcElement (filling element; for checking openings of selected elements) |
| Supported | Size of aggregates   * *Each IfcOpeningElement shall only have zero or one filling.* |
| Link | <http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-104> |

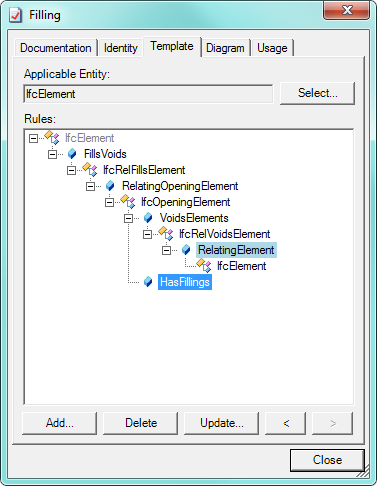
**Example-CV104.xml** (with modified concept template “Filling”)

Concept Template: Filling elements like door and window are typically used as root objects as they are more important than the opening element itself. The concept template “Filling”, which is part of the IFC4 documentation, is using the direct relationship *IfcRelFillsElement.RelatingOpeningElement* instead of its inverse relationship *IfcOpeningElement.HasFillings.* It therefore does not enable to restrict the number of filling elements. There are two solutions to deal with that issue:

1. modify “Filling” by adding a *HasFillings* attribute rule with cardinality setting “Zero-to-One” to the *IfcOpeningElement* entity rule as shown in the figure below or
2. add a new concept template to be used by *IfcOpeningElements* as root concept.

The second solution is more general and probably more clear but may requires to add additional restrictions if for instance not all types of filling elements shall be supported. Also, if not all opening elements shall be part of an exchange requirement, maybe only openings of supported building elements, then it is not recommended to use *IfcOpeningElements* as root concept.

NOTE: In case of having inverse relationships it is normally sufficient to add only one direction.



Concept: If using the concept template “Filling”, then all filling elements like *IfcDoor* and *IfcWindow* must be used as root concept. If a global check of all openings is of interest, then the new concept template must be used by *IfcOpeningElement* as root concept. In both cases no additional template rules are required.

### #CV-2x3-106: agreed use of geometric representation context and sub context

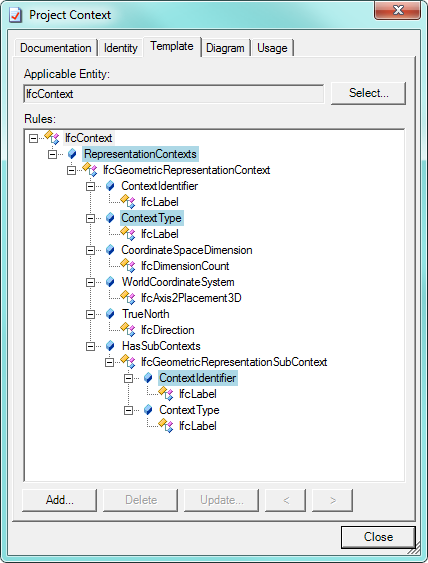
|  |  |
| --- | --- |
| Root | IfcProject |
| Supported | Size of aggregates   * *each IFC model shall have a maximum of 2 (and minimum of 1) instances of IfcGeometricRepresentationContext*   Content of simple data type values   * *an IfcGeometricRepresentationContext with ContextType = 'Model' shall be included …*   Conditions   * *… it shall have a minimum of one sub context, being IfcGeometricRepresentationSubContext.ContextIdentifier = 'Body'* |
| Not supported | - |
| Link | [http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-106](http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-100) |

**Example-CV106.xml**

Concept Template: The used concept template “Project Context” defines three parameters, (1) *RepresentationContext*, (2) *ContextType* and (3) *SubContextIdentifier*. To be more flexible the number of *IfcGeometricRepresentationContext* instances is not restricted on concept template level using the *Cardinality* attribute. This is done on concept level using the rule grammar.

The concept template includes references to *IfcGeometricRepresentationSubContext*, which is a subtype of *IfcGeometricRepresentationContext*. Although defined for *IfcGeometricRepresentationContext* all required data like *ContextIdentifier* or *ContextType* must also be defined for *IfcGeometricRepresentationSubContext* because it has a different meaning than context instances that are directly attached to *IfcProject*.

NOTE: *IfcGeometricRepresentationSubContext* not includes *CoordinateSpaceDimension*, *WorldCoordinateSystem* and *TrueNorth* as they are derived attributes.



Concept: One concept of above shown concept template is defined for *IfcProject*. It is defined to be mandatory for the exchange requirement “Examples”. It also adds a template rule that restricts maximum cardinality of *RepresentationContext* and defines allowed values for *ContextType* and *SubContextIdentifier* using the following expression string:

RepresentationContext[Size]<3 AND   
ContextType[Value]='Model' AND   
SubContextIdentifier[Value]='Body'

NOTE: The “Less Than” operator < must be escaped by &lt; in the XML file.

### #CV-2x3-112: support for CSG geometry is required on import

|  |  |
| --- | --- |
| Root | all IfcProduct elements |
| Supported | Conditions + Type of referenced data   * *The items within the Items list of IfcShapeRepresentation (being an 'CSG' representation) shall be of type IfcBooleanResult.* *The Items list shall not include IfcCsgSolid, or IfcPrimitive3D.* * *Within an IfcBooleanResult an IfcHalfSpaceSolid shall only be used as the SecondOperand of an Operator=".DIFFERENCE." (and not for an Operator=".UNION." or ".INTERSECTION.").* |
| Not supported | <TBD> |
| Link | [http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-112](http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-100) |

Proposed solution: Make use of partial concept templates as suggested in section 3.2 to deal more efficiently with different types of geometric representations and to enable recursion.

Concept Template: It is suggested to define a *ConceptTemplate* for boolean result CSG geometry that uses a partial concept template for *IfcBooleanResult*. This partial concept template shall define parameters for *IfcBooleanResult.FirstOperand* and *IfcBooleanResult.SecondOperand* so that supported types can be configured by the concept. For all supported geometric definitions, including *IfcBooleanResult* ~~but except for~~ *~~IfcCsgSolid~~* ~~and~~ *~~IfcPrimitive3D~~,* an *EntityRule* is defined that sets a link to a partial concept template with further details.

Concept: To IfcProduct entities a concept for CSG geometry is added, which excludes all not supported CSG types by defining a constraint for *First-* and *SecondOperand*.

### #CV-2x3-116: no use of subtypes of IfcStyledItem for assigning presentation information

|  |  |
| --- | --- |
| Root |  |
| Supported | Type of referenced data (Type 3)   * *The link between the presentation and the geometry or material definition shall always been made by IfcStyledItem not by its subtypes:* |
| Not supported | <TBD> |
| Link | NOTE: Not relevant for IFC4 as there are no subtypes anymore.  [http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-116](http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-100) |

Proposed solution: Make use of the attribute *EntityRule.IncludeSubtypes* as suggested in section 3.4 (Link to *IfcStyledItem* + *IncludeSubtypes = FALSE*)

Concept Template:

<TBD>

Concept:

<TBD>

### #CV-2x3-120: material information for decomposed elements shall only be given at the element part level

|  |  |
| --- | --- |
| Root | IfcBuildingElement (and subtypes) |
| Supported | Conditions + Existence of attributes and references   * *If the building element is a container, then the material information (IfcRelAssociatesMaterial --> IfcMaterial| IfcMaterialLayerSet| IfcMaterialLayerSetUsage) shall only be assigned to the parts, not to the container* |
| Not supported | Use of parameters from different concept templates |
| Link | <http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-120> |

Concept Template:

<TBD>

Concept:

<TBD>

### #CV-2x3-121: decomposed elements shall have a maximum of 1 level decomposition depth

Outdatet agreement, but interesting as an example!

|  |  |
| --- | --- |
| Root |  |
| Supported | Nesting and recursion   * *The parts contained within an element container shall not be containers by themselves. I.e. the decomposition hierarchy shall only have 1 level depth.* |
| Not supported |  |
| Link | [http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-121](http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-120) |

Concept Template:

<TBD>

Concept:

<TBD>

### #CV-2x3-142: agreement on having at least one instance of IfcBuilding as part of the spatial structure

|  |  |
| --- | --- |
| Root | IfcProject, IfcSite |
| Supported | Existence of elements (global and local)   * *An IFC exchange file conforming to the coordination view shall have: minimum of one instance of IfcBuilding within the spatial structure, either directly assigned to IfcProject, if no IfcSite is present, or assigned to IfcSite.* |
| Not supported |  |
| Link | <http://www.buildingsmart-tech.org/implementation/ifc-implementation/ifc-impl-agreements/cv-2x3-142> |

Concept Template:

<TBD>

Concept:

<TBD>

## “Model Checking” use-cases from the Dutch AEC industry

by Léon van Berlo, comments by Matthias Weise

This chapter describes some “Model Checking” Use-cases from the Dutch AEC industry. These use cases are now realized with Solibri, Tekla BIMSight or other custom build software tools. The goals of this chapter is to inspire mvdXML developers (or any other developer) to improve the MVD standard.

### Use case 1: Psets

Are all Psets filled in, and are they correct?

### Use case 2: Masonry supplier

Is the type of masonry bond added to the wall as a property (and does the value come from a predefined enumeration).

### Use case 3: Fire resistant doors

Are all doors between compartments of the building fire resistant? And is the fire rating in minutes?

### Use case 4: Is there a door?

Does every space (IfcSpace), except service shafts, have a minimum of one door (IfcDoor)?

### Use case 5: Is an object on the storey it should be?

Check if (/how) an IFC object is related to an IfcBuildingStorey. Then check if the highest geometrical and lowest coordinate of the object. The difference between these coordinates is the height of the object. This should be lower than the floorheight of the related floor.   
This use case is to check if facades and walls are split up between storeys rather than crossing over multiple storeys.

### Use case 6: Elevation

Check if the absolute elevation of an object is between the lower and upper boundaries of the related buildingstorey.

### Use case 7: Material

Do all objects (IfcProducts) have materials?

### Use case 8: classification consistency

Does the classification code (Uniclass, omniclass, NlSfb, etc.) match the IFC object? For example, is an object with an OmniClass specification ‘Door’ actually an IfcDoor in the model? Or a Dutch example, is an object with classification code ‘21’ actually an IfcWall in the model?

### Use case 9: program requirements

Does the design (/model) comply with the program requirements? For example, do all spaces meet the minimum surface area and volume? Are the toilets not located too far away from the meeting rooms?

### Use case 10: Breps versus extrusions

Are there any Breps or extrusions in the geometry of the model?

## Validation testing for the Precast BIM Standard

# Issues and Agreements

This chapter describes agreements how to apply mvdXML where current specification in unclear or ambiguous.

## Metric [Size] for AttributRule

Short description of the relevant mvdXML specification:

* The attribute *RuleID* of *EntityRule* and *AttributeRule* defines the parameter that can be used in the rule grammar, specified by *Constraint.Expression* or *TemplateRule.Parameters*.
* *TemplateRule.RuleID* sets a link to an *EntityRule* (or an *AttributeRule*) to differentiate between outer and inner parameters (see mvdXML 1.0)
* Outer parameters define a condition
* Inner parameters define a constraint

The following figure shows that if *EntityRules* can be linked from *TemplateRules* only then *AttributeRule* “A1”, which is the first rule attached to *ConceptTemplate*, cannot be constrained by a *TemplateRule.Parameter*. The parameter “A1[Size]<2” would define a constraint as it is an outer parameter when linking to EntityRule E1.

Therefore, it is suggested to allow links from *TemplateRule* to *AttributeRule* also.



## Partial ConceptTemplates

**Request:** *ConceptTemplate* definitions shall be usable by other *ConceptTemplate* definitions. This would enable:

1. to reduce specification and maintenance work (universal sub structures must be defined only once) and
2. to specify dependencies (link to required structures that are defined in more detail somewhere else)
3. to deal with recursive concept template definitions (e.g. IfcBooleanResult that use other IfcBooleanResult instances as operands)

**IFC examples:** Definition of a vertex point or polyline could be defined in an own partial concept template to be reused by higher-level concept templates like product shape, placement or grid. This would reduce specification work. Another example for product shape definition is to set a link to a concept template that defines the geometric representation context but is attached to another root concept (in this example *IfcProject*).

**Suggested solution:** The following requirements must be fulfilled by mvdXML to deal with above examples:

1. *ConceptTemplate.ApplicableEntity* must allow to select any entity (not only root entities) to define partial templates for non-root entities like *IfcVertexPoint*.
2. *EntityRule* should enable to set a link to *ConceptTemplate* in order to be able to use partial templates or select the *ConceptTemplate* that further defines the usage of this entity.

Requirement (1) can already be handled by the mvdXML 1.1 schema. But the documentation must be updated so that not only root entities can be used for the definition of *ConceptTemplates*.   
Requirement (2) requires to add a new attribute to *EntityRule*. It is suggest to extend the current definition by the optional attribute *PartialConceptTemplate*, which defines a link to *ConceptTemplate*.

The following clarification is proposed for *ConceptTemplate*:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| … |  |  |
| applicableEntity | String  [0:?] | Indicates the *~~IfcRoot~~*~~-based~~ entities, including all derived entities, for which the concept applies. It is recommended to use a single base class (e.g. *IfcElement*). This value provides the context for any attribute rules and is used within MVD tools to filter the list of available templates for particular entities. For a sub-template, the applicable entity must be the same type or a subtype of the outer template. This value may be blank to indicate an abstract template that cannot be instantiated, containing sub-templates for specific entities. |
| … |  |  |

The following extension is proposed for *EntityRule*:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| … |  |  |
| PartialConceptTemplate | ConceptTemplate [0:?] | Link to *ConceptTemplate* that provides further details about how to handle that entity. If *EntityRule.EntityName* does not match with *ConceptTemplate.ApplicableEntity* (*ApplicableEntity* is a root entity), then this entity will be managed by other root entities (may be defined by a list of concept templates)*.* Otherwise it is a partial template that extends this entity definition. |

## Abstract ConceptTemplates

**Request:** Support “wild card” attribute definitions to be able to add rules that are applicable for subtypes only. This would be interesting for instance for defining a “Type” template for *IfcObject* that not only includes the attribute “ObjectType” but also “PredefinedType” that is defined by subtypes of *IfcObject*.

**Solution:** No changes required – see notes for *ConceptTemplate.Rules*, which also applies to *EntityRule.AttributeRules*.

## Interpretation of ConceptTemplate for generation of subset schemas

**Request:** For the generation of a subset schema it is not clear which entities of a *ConceptTemplate* should be pulled in a subset schema.

The following options are of interest:

1. Selected entity types shall be added to the subset schema, which is in particular interesting and required for adding non-root entities.
2. Selected entity types are supported but shall not be added to the subset schema as it is expected that they will be selected by other definitions. This option is in particular interesting for root entities.

For selection of supported entity types the following options are possible:

1. Selected entity types only
2. Selected entity type + subtypes

**Solution:** It is suggested to add two new flag attributes to EntityRule to clarify selection of entities.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| … |  |  |
| IncludeSubtypes | Boolean | Defines if subtypes of this entity are selected or not. Default value is TRUE. |
| AddToSchema | Boolean | This setting is needed for the generation of the subset schema. It defines if all selected entities shall be included in the subset schema or not. Default value is TRUE. |

For clarification of the current agreement for *AttributeRule.EntityRules* an empty list should be avoided. An empty list would use the entity (or entities in case of SELECT data types) as defined in the underlying schema with the settings *IncludeSubtypes*=TRUE and *AddToSchema*=FALSE.

## Exclude subtypes from referenced entities

<TBD>

## Using unnamed INVERSE relationships

<TBD>

# Discussions

## Configuration of min and max cardinality

[Chi Zhang]:

Is it possible to make the cardinality also definable on the Concept level (overrides the definition in the ConceptTemplate)? If the cardinality can only be defined in the ConceptTemplate, when we want to change the cardinality of some Concept, we have to define a new ConceptTemplate (or maybe a SubTemplate, but the current version of ifcDOC does not support this little change on SubTemplate, we change it by hand), but is that the original thought of ConceptTemplate (should the ConceptTemplate be more general, may I ask)?

[Matthias Weise]:

On concept level you could use the metric [Size] for this. However, I would expect cardinality constraints on the concept template level. For instance you could have a ConceptTemplate for Polygon with min=3..max=n points. A Rectangle would have min=max=4 points. But as mentioned by you, it should be possible to define Rectangle as a sub-ConceptTemplate of Polygon. So, redefinition of cardinality should be possible for sub-ConceptTemplates (if it is more restrictive).

Could you explain your example where you want to redefine cardinality on concept level?

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Tools and Reference Implementations

<TBD if of interest>

## IFC Documentation Generator

## Constructivity Viewer

## GTDS

## BIMServer

# Appendix

<Table from Donghoon to be added – see email from 26.11.2013>