www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

INTEROPERABLE VALIDATION FOR IFC BUILDING MODELS USING OPEN STANDARDS

SUBMITTED: November 2014
REVISED: December 2014
PUBLISHED: January 2015 at http://www.itcon.org/2015/2
GUEST EDITORS: Mahdavi A. & Martens B.

Chi Zhang
Department of Built Environment, Eindhoven University of Technology, Eindhoven, The Netherlands;
c.zhang@tue.nl

Jakob Beetz
Department of Built Environment, Eindhoven University of Technology, Eindhoven, The Netherlands;
j.beetz@tue.nl

Matthias Weise
AEC3 Deutschland GmbH, Muenchen, Germany;
mw@aec3.de

SUMMARY: *In this paper we are reporting on a prototypical implementation of a model view checker for model instance validation of Industry Foundation Classes (IFC) models. This checker is developed based on the open standards mvdXML as the format for structuring validation rules and the BIM Collaboration Format (BCF) to issue reports as a result of the checking process. The checker is implemented on top of the open source bimserver.org framework. The research presented here has two main aims: (1) to develop an open source IFC validation tool based on flexible and standardized method; (2) to identify issues and capabilities of the current mvdXML rules based on real-world scenarios and to develop stable and easy-to-use IFC validation methods using open standards. Two BIM operational standards required by local building regulations and laws, the Dutch Rgd BIM Norm, and the Norwegian Statsbygg BIM Manual are used to validate both the mvdXML standards' capabilities and the tools implementation. The rules from these standards are categorized into different rule types and converted to mvdXML templates and rules. These rules are then tested using a prototypical, open source software tool. By combining this tool with a BCF server we demonstrate the deployment of such automated checking procedures in real working processes. Based on these experiences, a detailed discussion about identified issues is provided as the starting point for the future research and a feedback to standardization organizations.*

KEYWORDS: *Industry Foundation Classes, Model View Definition, Constraint, Validation, Use-case, mvdXML, BIM Collaboration Format.*

REFERENCE: *Chi Zhang, Jakob Beetz, Matthias Weisen (2015). Interoperable validation for IFC building models using open standards. Journal of Information Technology in Construction (ITcon), Special Issue: ECPPM 2014, Vol. 20, pg. 24-39, http://www.itcon.org/2015/2*

1. INTRODUCTION

The building industry is a collaboration environment that requires repeated, iterative data exchanges and communication among different domains and applications in a high frequency. To automate information processing, standardized and qualified data is necessary for efficient working processes. As the design and construction processes become more and more complex, traditional information mediums such as paper-based documents cannot provide the required integrity, precision and timeliness. Many researches have suggested that this objective can be approached by using vendor-neutral and open building information models such as the Industry Foundation Classes (IFC) to capture and exchange data (ISO 16739 2013; Eastman et al. 2011; Berlo et al. 2012). Various systems can implement converters between the IFC schema and their native data models to export and import IFC instances to exchange information with each other. This approach has contributed to addressing the problem of data exchange and integration. In addition to the shared IFC data model itself however, the interoperability also depends on two other factors: a) The quality of implementations of IFC export-import converters; b) The quality of the instance modelling of buildings. The respective work is carried out by application implementers and domain end-users alike. To evaluate their work in order to get reliable, high-quality and interoperable data is crucial for improving the efficiency of working processes. While the quality assurance of software implementations is subject to the certification processes by the buildingSMART standardization organization (Hausknecht et al. 2014), the quality of instance model produced with (certified) software tools by end-users has to be validated in a day-to-day practice during all lifecycle stages of a building.

1.1 Interoperability issues of the Industry Foundation Classes

The Industry Foundation Classes is a standard data model that supports a full range of data exchanges among heterogeneous applications. Its schema is developed in the EXPRESS modelling language (ISO 10303-11 1997). Like other general data models, IFC has provided a rich set of modelling constructs to capture data in order to be compatible with different domains and task scenarios. It has been structured into four conceptual layers, which in total contain about 800 entity definitions, thousands of attributes and even more standardized properties augmented to the model schema in external property sets to represent information (buildingSMART 2013a). This extent makes it one of the largest EXPRESS-based data model used in practice to date. The richness of the full IFC model constitutes a threshold for particular exchange tasks and can hardly be fully supported and implemented by many specialized domain applications (IUG 2012). On the other hand, a wide range of detailed and specific domain information are not explicitly modelled and covered on a schema level. Moreover, many strict data modelling mechanisms on the schema level have side effects that do not meet the flexible requirements in daily information processing practice. An example is the strict attribute inheritance that would mandate information often too detailed to be provided e.g. in early project phases.

To balance these dilemmas, there are very few data constraints designed in the IFC standard to specify which information should be exchanged and how it should be modelled in instances (Venogopal et al. 2012). On the schema level *OPTIONAL* and weakly typed attributes are the dominant citizens, and there are very few constraints requiring existence of specific entities. Object instances can also be semantically extended by mechanisms such as *IfcProperty* and external classification references. EXPRESS and thus IFC also allows to capture the same information in different ways. All of these features have provided the flexibility required by different use-cases but also give undesirable freedom for application implementers and domain end-users. As a result, a syntactically correct IFC instance might e.g. miss needed information. For example, according to the schema an *IfcDoor* has only two mandatory attributes: *GlobalId* and *OwnerHistory*, which are inherited from *IfcRoot*. Other domain specific information is unnecessary for a door object to be syntactically valid. In practice however much more information is required for many tasks. During the design phase information about geometry, location and material must be provided, while procurement processes demand door types, price and manufacturer information. Before exporting to the IFC format, designers or engineers have to properly model such information into building models. When models are converted from their native formats into IFC, information will be translated to corresponding IFC concepts: a door material is converted to an *IfcMaterial* structure referencing the *IfcDoor* instance. This entire process highly relies on knowledge, personal experiences and interpretations of domain experts and implementers, which are not reliable enough in a computational environment. Some experiments showed that various applications have different approaches how buildings should be modelled and mapped to the IFC schema and what information is required, which in real practices cause additional remodelling and plenty of manual work (Jeong et al. 2008; Lee 2011; Belsky et al. 2013).

Increasing specialization and automation in the building industry require high levels of interoperability. For many applications we must make sure that the data needed for these processes is contained in models with proper representations by types, properties and names (Eastman et al. 2009). Such validated IFC building models are an important pre-condition for executing many automated tasks including building performance analysis and code compliance checking.

1.2 Model view and IFC validation

A semantically validated IFC instance should not only comply with all the syntactic constraints defined in the schema, but also has to fulfil additional rules about the proper usage of model elements (Yang and Eastman, 2007). Thus, which rules should be defined and how to define and apply them are the essential challenges. The standard methodology of the Information Delivery Manual (IDM) and Model View Definition (MVD) has been introduced from the working process point of view to define required information for particular scenarios (Wix 2005; NBIMS 2007; IUG 2012; Karlshoej 2012). In the first step, an IDM defines use-case scopes and working process models, according to which the possible information needs and constraints are also documented in text references as exchange requirements and business rules. These plain text requirements will be structured to exchange requirement models and then bound with specific IFC entities, attributes and types as MVDs. These MVDs reduce the scope of the full IFC model to subsets, which should be supported by the export-import routines of related applications. They also declare semantics of specific IFC model structures as the implementation agreements between two or more parties. For example, the concept of "door types" in all applications will be converted to the *IfcDoorType* connected to the *IfcDoor* by the *IfcRelDefinesByType* structure instead of other constructs e.g. *ObjectType* attribute of *IfcDoor*. Following this methodology, various groups and consortia have developed IDMs and MVDs for their target domain model exchanges (buildingSMART 2011). These common MVDs such as the Coordination View have a universal scope and are not specialized to regional or project-specific requirements.

A number of national, company- and project-specific BIM standards and agreements have been developed independently from this technological approach. The Dutch general service administration (Rijksgebouwendienst - Rgd) BIM Norm in The Netherlands and Statsbygg BIM Manual in Norway are just two of a growing number of national standardization efforts (Rillaer et al. 2012; Statsbygg 2011; BIM Guides Project 2014). These IFC-based BIM standards can be regarded as sets of business rules, which define requirements that IFC instances shall fulfil in specific contexts. The guides are used to (1) evaluate export-import capabilities of BIM applications and to (2) check the semantic integrity and correctness of models made by end-users. Typically they specify entities and properties that must be included, their classification references, naming conversions and value range constraints. For example, "spaces shall be modelled with 3-dimensional space objects (*IfcSpace*)" (Statsbygg 2011).

The MVD and BIM standard efforts focus on the development of domain-specific concepts and requirements but neglect the definition methods for describing them. In order to efficiently apply them, these documents must be translated to computer interpretable formats to facilitate automated processing. Currently, this process is supported by a limited set of tools. Many MVDs are documented in text or table based formats (BLIS-project 2009) intended for human readers only. Although some checking methods and commercial tools have already been implemented, most of them are based on proprietary rule definitions or closed hardwired procedural code and thus cannot be flexibly accessed, reused and edited by end-users.

The building industry is a flexible field which often needs many customizations for information requirements. Hence, a validation tool based on an open and standardized method which supports reusing rules is required.

To address the issues mentioned above and elaborated in section 2, we are reporting on an open source model view checker based on open standards to validate IFC building models in this paper. We show a prototypical implementation and test it with example use cases. In the section 2, we give a brief review of related work which contributes to the IFC validation. The overall methodology and essential implementation details of this checker are introduced in section 3. In section 4, use-cases from two BIM standards are categorized into different rule types and presented with examples. In section 5, a transformation of the checking results into the form of Building Collaboration Format (BCF) issues are presented, and a workflow for using this these tools in real-world scenarios is proposed. Sections 6 and 7 provide a conclusion of the test use-cases, and a detailed discussion about identified issues and future developments. The aim of this research is not only to implement an open source IFC validation tool, but also to build the foundation of developing stable and easy-to-use IFC validation methods.

2. RELATED WORK

Both MVDs and BIM standards provide additional rules for IFC validation. MVDs focus on extracting integral model subsets for IFC implementation purposes, whereas BIM guides are more fragmented requirements applied to check IFC instances. In this section, we provide a brief review of model view definition methods and current implemented methods and tools for checking IFC models.

2.1 Model View Definition methods

An MVD is an implementation reference for application developers. The introduced methods in this subsection overlap with each other but have different foci. A comparison about their functionalities and differences is presented in Table 1.

The mvdXML released by buildingSMART is an open standard to define model subsets and validation rule-sets. (Chipman et al. 2013). An mvdXML file is an XML instance that adheres to the mvdXML schema, so it can be developed by the official IfcDoc tool as well as common XML editors (Chipman 2012). The purposes of mvdXML are (1) to limit IFC scopes to subsets, (2) to generate MVD documentations and (3) to define validation rules. Some of the technical details of mvdXML are elaborated on 3.1.

The eXtended Process to Product Modeling (xPPM) is a tool that provides an integrated solution for the IDM –MVD process (Lee et al. 2013). In comparison to other MVD methods, xPPM is more oriented at the IDM process, applying a subset of the Business Process Modeling Notation (BPMN) (OMG 2011), which is also the standard process modelling method in IDM. The xPPM covers all IDM stages and can also be used to bind functional parts (exchange requirement concepts) with IFC elements as MVDs (Lee et al. 2013). Following the internal defined schemas, developed exchange requirements and functional parts are maintained in XML formats, which can be used to generate documentation of IDMs and MVDs. Some constraints like UNIQUE and cardinality are possible to add, but they are not practically used to validate IFC instances.

The Generalized Model Subset Definition (GMSD) is an EXPRESS-based meta-schema constructed to develop view definitions and object selection with special attention to IFC (Weise et al., 2003). It has provided a more straightforward solution to define a model subset as a composition of a set of entities, types and attributes from a product model schema. This method is implemented on the ViewEdit (Katranuschkov et al. 2010, Windisch et al. 2012), which is a schema-level model filter developed to generate and edit GMSD instances to extract model subsets from the IFC schema.

TABLE 1. Comparison of MVD development methods. X = yes/supported /major purpose; O = partly supported/minor purpose; - = no/not supported/not exist

	mvdXML	xPPM	GMSD	SEM
IDM	-	X	-	-
Model subset extraction	X	X	X	X
Validation rules	X	O	O	O
Implementation of the validation function	-	-	-	-
Open standard	X	-	-	-
Edit tool	IfcDoc, XML editors	xPPM	ViewEdit	Model View Developer plugin

The Semantic Exchange Module (SEM) is an on-going research focusing on reusability of MVDs in order to approach efficient and consistent IFC implementations (Venugopal 2011). It provides a module-based development method to keep concept consistency and prevent redundancy among different MVDs. A SEM is similar to a model view concept, but defined as a binding not only to a set of IFC elements, but also to the corresponding set of native model structures of domain applications (Eastman et al., 2011). This method has been implemented in the research prototype of Model View Developer plugin (Venugopal 2011).

2.2 Validation approaches and tools

In order to check models, text-based requirements and rules have to be converted to computer-executable rule-sets. A detailed review about current applications of rule checking is conducted by (Eastman et al. 2009). Although this work focuses on code compliance checking, some of the introduced platforms can also be applied for validation purposes. In the following review, we reference some conclusions of (Eastman et al. 2009). The existing implemented approaches can be roughly categorized as programming methods and schema-based methods.

2.2.1 Programming methods

All the programming languages can be considered as computer executable rule formats to check building models. Hard-coding rules into high-level imperative programming languages such as C++ and Java is currently the most widely used approach for checking models that is also commonly implemented in commercial checking platforms. The Solibri Model Checker (SMC) is now one of the most widely used checking applications (Eastman et al. 2009). SMC is a powerful checking platform that can also be used to perform e.g. clash detection and code compliance checking. It has already pre-converted some standardized model views and BIM manuals to rule-sets, such as the BIM Coordination View. Other, customized rule sets such as the Dutch Rgd BIM Norm are being implemented upon request. It can also be customized further by some rule configuration tools e.g. its built-in Rule Manager. However, these requirements or rule-sets are implemented with proprietary definition means, and the customization is mostly limited to pre-defined parameters in these hard-wired rules. Moreover, developed rule-sets are dependent on specific platforms and thus cannot be adopted by other applications.

2.2.2 Schema-based methods

The EXPRESS language provides schema-level constraint mechanisms such as *WHERE* and *UNIQUE* rules to restrict instance models. During the last updates and releases of the IFC schema, an increasing number of rules have been introduced. For example, the IFC4 schema has defined new rules like "an *IfcDoor* should only be typed by *IfcDoorType*", which are not contained in IFC2X3. In the IFC implementation phase, these rules can be interpreted by a few IFC toolboxes that understand the EXPRESS language to generate schema-compatible instances (buildingSMART 2013b). In the STEP standard, the EXPRESS-X (ISO10303-14 1999) is provided as an extension of EXPRESS includes extra constructs to map model views and define additional constraints. EXPRESS and EXPRESS-X can be interpreted by some IFC validators such as the one built into the Jotne EDM Model Server to check instance models against the schema.

Another example based on this method is the IfcCheckingTool, which is a small-sized tool dedicated for IFC validation. It is now a component in the official IFC certification platform in buildingSMART (Hausknecht et al. 2014). It is a C++ based tool with rules defined in DLLs by ECCO toolkit, which is a commercial software development environment for EXPRESS and EXPRESS-X based applications. Currently the only open tool implementing schema-based ruled validation is the jSDAI by framework LK Software licenced under the AGPL v3 model.

2.3 Conclusion

Existing model view definition methods including GMSD, xPPM and SEM mainly focus on e.g. generating MVD documentation or modularizing model views, but are not primarily aimed to define computer executable business rules to check IFC instances. Some checking platforms and tools have implemented programming methods to check models, but these "black box" methods cannot be fully accessed by domain end-users. The schema-based methods provide an open environment for rule development. As ISO standards, however EXPRESS and EXPRESS-X are actually not popular languages even among software engineers, and current implementations of these methods are still limited mainly on commercial platforms. All the existing methods and tools cannot provide an open and low-cost rule checking environments. This is particularly problematic for SMEs that often do not have the resources to adapt these demanding technologies. MvdXML is currently the only open standard dedicated for model view definition and IFC validation. An implementation for mvdXML rule checking is thus highly desirable for research and develop communities as well as for end-user practitioners.

3. IMPLEMENTATION APPROACH

Generally, three steps are needed in the IFC validation process: (1) interpretation of requirements and structuring validation rule-sets, (2) execution of the checks and (3) report generation. This implementation is based on the open source bimserver.org framework (Beetz et al., 2010), integrating two open standards—mvdXML (Chipman et al. 2013) and the BIM Collaboration Format (BCF) (Stangeland 2011) respectively as the validation rule-sets and issue reports. Basically, the IFC instances and mvdXML files are the input of the checker while sets of BCF files are the output (Fig. 1).

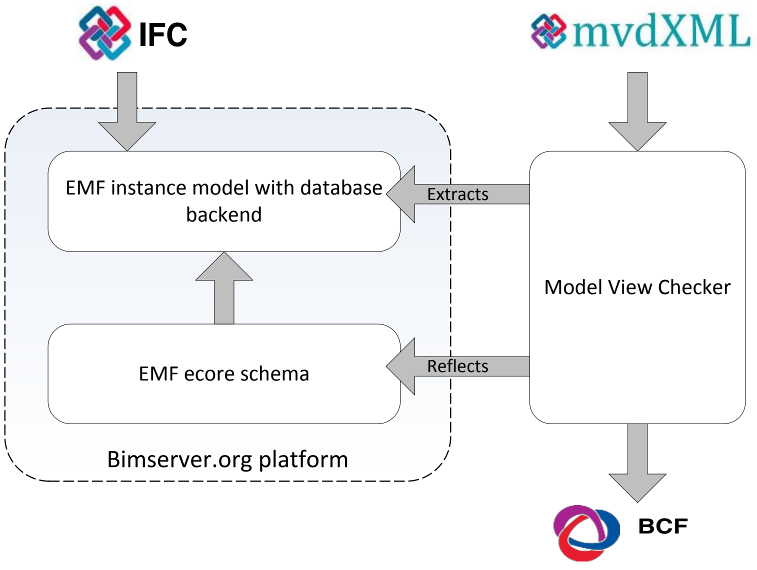


FIG. 1: General architecture and interaction between this checker and bimserver.org

3.1 Development of model view rule-sets

This implementation is based on the 1.1 version of the mvdXML standard. Its detailed specification can be found in (Chipman et al. 2013). In mvdXML, the concepts and rules are described on *Concept Template* and *Concept* levels. Every *Concept* refers to a *Concept Template* as its basic structure. The outcome of the definition can be considered as a set of concepts, each of which has a tree structure from the root entity (an IfcRoot subclass) to leaf nodes (attribute values, referenced entities). Additional constraints can be defined for the root entity's attributes or recursively for the referenced entities' attributes. Depending on "mandatory", "optional" or "excluded" for different *Exchange Requirements*, it defines the rules that every object of the root entity should follow. The 1.1 version also provides a machine-readable rule grammar which can be implemented with e.g. ANTLR or other parser generators (Parr, 2007). It can be used to define more sophisticated rules either in *Concept Template* or *Concept* nodes to enhance the expressivity of the entire definition. A rule like "every *IfcWall* should be typed by an *IfcWallType*" in IFC4 is defined as illustrated in Fig. 2. The *Concept Template* in this case can be defined as a basic association structure from *IfcObject* to *IfcTypeObject* with additional cardinality constraints on *IsTypedBy* attribute. The "Concept" which refers to this template is "mandatory" to be applied on each *IfcWall* instance. On the *Concept* level, the type of value can also be specified further by the additional rule to say that the attribute of the *RelatingType* should be an *IfcWallType* (Fig. 2).

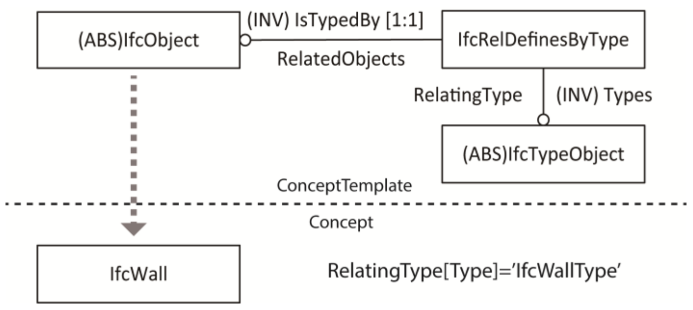


FIG. 2: Example Model View Concept in mvdXML

3.2 Check execution

In an IFC instance file, the information is provided by the attributes of all existing object instances. The bimserver.org platform already provides a convenient API to extract attribute values. In this platform, the EXPRESS schema of IFC has been converted to an Eclipse Modelling Framework (EMF) which is used to generate corresponding Java classes for IFC entities and types (Beetz et al. 2010). By this mechanism, related IFC objects in instance files and their attributes can be extracted by their names defined in the mvdXML file. Depending on rule types in mvdXML, these values are checked to evaluate whether their existence, quantities, contents, uniqueness and conditional dependencies fulfil requirements or not (Weise 2014). For every instance of every root entity defined in mvdXML, the rules and constraints should be evaluated to be true. For example, in the example illustrated in Fig. 2, every *IfcWall* (including its subtype *IfcWallStandardCase*) will be checked if it has the related *IfcRelDefinesByType* which has the *RelatingType* of an *IfcWallType*. If an object instance violates one rule, the program will generate an issue associated with the *GlobalId* of this root object, which, in this case, is an *IfcWall* object.

3.3 Report generation

In this implementation, we use the BIM Collaboration Format (BCF) to report identified issues. BCF is an open standard originally proposed by Solibri and Tekla to enable workflow communication between different applications (Stangeland 2011). Unlike traditional text-based reports, this method links communication to IFC building models, and it also can be easily implemented by visualization applications. It is currently endorsed by a number of software vendors, and is going to become an official buildingSMART specification.

After the checking, every generated issue will be captured in the form of a BCF report which mainly includes a markup file and a viewpoint file. The generated issue comments are contained in the markup file, which also contains the description of the "Concept" defined in the mvdXML file to make domain end-users understand the requirement that was violated. The viewpoint file defines view point cameras for issued objects. Based on the bimserver.org built-in IfcOpenShell library, which converts implicit geometric data in IFC instances to global coordinates, this checker derives bounding boxes of issued objects taking into account of aggregation relationships between objects. Viewpoint cameras are set up based on bounding boxes to generate snapshots of objects that triggered the issues. For this communication purpose, we suggest that the root entity in mvdXML should be defined as a tangible entity (*IfcProduct* subtypes) if possible.

4. USE-CASES

Real-world use-cases from the Rgd BIM Norm and Statsbygg BIM Manual are used to define rule-sets to test the checker. The validation rules can be categorized as follows: (1) checking data existence and cardinality, including existence of attribute values and referenced entities, and size of collection data types; (2) checking content of values, including the value of simple data types and collection types; (3) uniqueness of values; (4) checking the if-then conditional dependency and consistency among them (Weise 2014). The type (1) usually accompanies with the later three, while (4) is based on the checking results of (1), (2) and (3). Except for some rules that are out of the scope of the current mvdXML standard (e.g. metadata requirements such as names of IFC models, or clash detection which needs additional computation), a brief overview for all the rule types in the Rgd standard is listed in Table 2.

TABLE 2. Rule category of Rgd BIM Norm

Rule Types	Requirements in Rgd BIM Norm
(1) data existence and cardinality	§2.1.1, §2.1.2, §2.1.4, §2.1.7, §2.1.8, §2.1.9, §2.2.6.1, §2.2.6.2, §2.2.6.4, §2.2.6.5, §2.2.7.1, §2.2.7.2, §2.2.7.4, §2.2.7.5, §2.2.7.6, §2.2.7.7, §2.2.7.8, §2.2.7.9, §2.2.7.10, §2.2.7.11
(2) data content	§2.1.2, §2.1.7, §2.1.8, §2.1.9, §2.2.6.2, §2.2.7.1, §2.2.7,2, §2.2.7.3, §2.2.7.5, §2.2.7.6, §2.2.7.7, §2.2.7.8
(3) data uniqueness	§2.2.6.4, §2.2.7.6
(4)conditional dependency	§2.1.4, §2.2.6.3, §2.2.7.4, §2.2.7.7, §2.2.7.11

Some of the requirements have multiple clauses that belong to different rule types, so there are some overlaps between rule types in this table. In the following subsections, for each type of rule, we give an example from these two standards with two logic formulas using terms from description-based rules and IFC schema respectively to specify its semantics and make a comparison.

4.1 Data existence and cardinality

Data existence is the most common rule type, which is usually the pre-condition for other rule types. It specifies whether a schema-level "OPTIONAL" attribute or relationship should exist or not.

For example, for the rule "A building contains at least one level." (Rgd §2.2.7.4), the logical representation can be written as (1).

$$\forall x(\mathrm{Building}(x) \supset \exists y(\mathrm{Level}(y) \wedge \mathrm{contains}(x, y))) \quad (1)$$

When this rule is mapped to the IFC Schema, this rule can be represented as (2) to cope with the general mechanism of IFC model to use objectified relationships to express relations between objects..

$$\begin{aligned}\forall x(\mathrm{IfcBuilding}(x) \supset \exists z(&\mathrm{IfcRelAggregates}(z) \\ &\wedge \mathrm{isDecomposedBy}(x, z) \\ &\wedge \exists y(\mathrm{IfcBuildingStorey}(y) \\ &\quad \wedge \mathrm{relatedObjects}(z, y))))\end{aligned} \quad (2)$$

In mvdXML, data existence is defined by the "mandatory" key word on the *Concept* level. Cardinality rules can be defined in the *Concept Template*, and it also can be further restricted in the *Concept* by the token "[Size]" in the rules formatted by the rule grammar. This concept can be structured by the *Concept Template* of the basic aggregation relationship in the IFC schema (Fig. 3). The cardinality "OneToMany" is defined for *IsDecomposedBy* and *RelatedObjects*. This template is applied to *IfcBuilding* as a *Concept*. The type of *RelatedObjects* can be further restricted to *IfcBuildingStorey* by specifying "RelatedObjects[Type] = 'IfcBuildingStorey'".

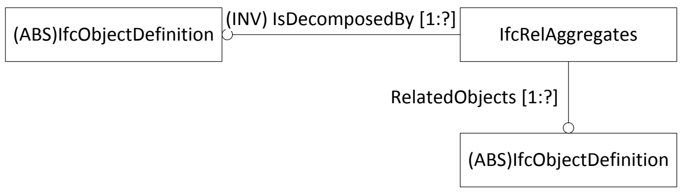


FIG. 3: Concept Template of the example rule in 4.1

4.2 Data content

There are some common scenarios of data content rules in these standards. For example, specific IFC objects (e.g. *IfcBuildingStorey*, *IfcClassification*) should follow some naming conventions; entities should be extended by specific properties; enumeration types should equal to some values. Rgd §2.2.7.8 defines a rule that an IFC object which provides access to a space, should have the additional property of "FireExit" within "Pset_###Common" (where ### is a placeholder for the specific object at hand, e.g. Door, Window etc.). If we take the door as an example of the access object, this rule can be represented as (3).

$$\begin{aligned}
&\forall x(\mathrm{Door}(x) \supset \exists y(\mathrm{hasPropertySet}(x, y) \\
&\quad \wedge \mathrm{Pset\_DoorCommon}(y) \\
&\quad \wedge \exists z(\mathrm{containsProperty}(y, z) \\
&\quad \wedge \mathrm{FireExit}(z))))
\end{aligned} \tag{3}$$

In IFC models, if the agreement is that this property should have the direct association with *IfcDoor* (not through *IfcDoorType*), this rule is represented as (4).

$$\begin{aligned}
&\forall x(\mathrm{IfcDoor}(x) \supset \exists y(\mathrm{IsDefinedBy}(x, y) \\
&\quad \wedge \mathrm{IfcRelDefinesByProperties}(y) \\
&\quad \wedge (\exists z(\mathrm{RelatingPropertyDefinition}(y, z) \\
&\quad \wedge \mathrm{IfcPropertySet}(z) \\
&\quad \wedge \mathrm{Name}(z, \text{" Pset\_DoorCommon"}) \\
&\quad \wedge \exists w(\mathrm{HasProperties}(z, w) \\
&\quad \wedge \mathrm{IfcPropertySingleValue}(w) \\
&\quad \wedge \mathrm{Name}(w, \text{" FireExit"})))))
\end{aligned} \tag{4}$$

In mvdXML, the data content rule type is specially defined by the "[Value]" token. It also has provided many operators e.g. "Equal" and "Greater_Than" and regular expressions to check the value. This example can be structured by the template illustrated in Fig. 4. This template is applied on *IfcDoor* instances. Two parameters in the template "PropertyName" and "PropertySetName" can be set up, respectively referencing the *Name* attributes of *IfcPropertySingleValue* and *IfcPropertySet*. On the Concept level, the additional rule is defined as "PropertyName[Value] = 'FireExit' AND PropertySetName[Value] = 'Pset_DoorCommon'" based on the rule grammar.

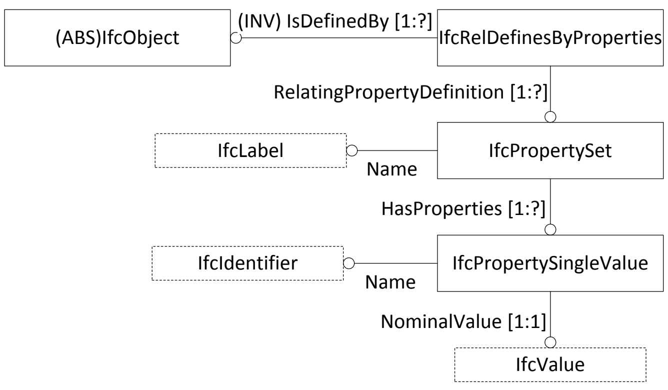


FIG. 4: Concept Template of the example rule in 4.2

4.3 Uniqueness

Only a very limited number of rules in these two standards demand the uniqueness of attribute instance values. Rgd §2.2.7.6 defines an implicit rule to specify that the space names should be unique. This can be represented as (5), in which we use isUnique(x) as a function which returns a boolean value.

$$\forall x(\mathrm{Space}(x) \supset \exists y(\mathrm{name}(x, y) \wedge \mathrm{isUnique}(y))) \tag{5}$$

This rule can be represented by the IFC elements in (6), in which we use predicate logic to represent the semantics of uniqueness. It can be read as "an IfcSpace x should have a name *y*, such that there is no IfcSpace z that when z is not x, its name w equals y".

$$\begin{aligned}&\forall x(\mathrm{IfcSpace}(x) \supset \exists y(\mathrm{name}(x, y)) \\ &\quad \wedge \neg\exists z(\mathrm{IfcSpace}(z) \wedge \mathrm{name}(z, w) \\ &\qquad \wedge ((z \neq x) \wedge (y = w))))\end{aligned} \tag{6}$$

In mvdXML, uniqueness rules are defined by the "[Unique]" token. In this case, a simple "Concept Template" can be structured (Fig. 5) and applied on *IfcSpace*. The attribute *Name* of *IfcSpace* can be defined as "Name[Unique] = TRUE".

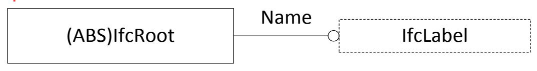


FIG. 5: Concept Template of the example rule in 4.3

4.4 Conditional rules

The conditional rule is to check the dependency or consistency between checking results of aforementioned types of rules. For example, Rgd §2.2.7 specifies that "each geometric building object is associated with the appropriate building level, taking into account the hierarchical relationship between IFC objects". This rule can be interpreted as "if an element is not composes of other elements, it should have an association with a building level" specified in (7). This rule is to check a consistency between the checking results of two data existence rules.

$$\begin{aligned}&\forall x(\mathrm{Element}(x) \wedge \neg\exists y(\mathrm{composes}(x, y)) \\ &\supset \exists z(\mathrm{BuildingLevel}(z) \wedge \mathrm{hasAssociation}(x, z)))\end{aligned} \tag{7}$$

The IFC version of this rule is specified in (8).

$$\begin{aligned}&\forall x(\mathrm{IfcElement}(x) \wedge \neg\exists y(\mathrm{Decomposes}(x, y)) \\ &\supset \exists w(\mathrm{ContainedInStructure}(x, w) \\ &\wedge \mathrm{IfcRelContainedInSpatialStructure}(w) \\ &\wedge \exists z(\mathrm{RelatingStructure}(w, z) \wedge \mathrm{IfcBuildingStorey}(z))))\end{aligned} \tag{8}$$

In the mvdXML standard, conditional dependency relationships are currently implemented by the logic connectors of "AND", "OR" and "XOR". In this case, the *Concept Template* can be set up as illustrated in Fig. 6 and then applied to *IfcElement* as a *Concept*. Parameters are defined for the attribute *Decomposes* and *ContainedInStructure*. The if-then condition in this case can be defined as: "(Decomposes[Size] = 0 AND ContainedInStructure[Size] = 1) OR Decomposes [Size]=1".

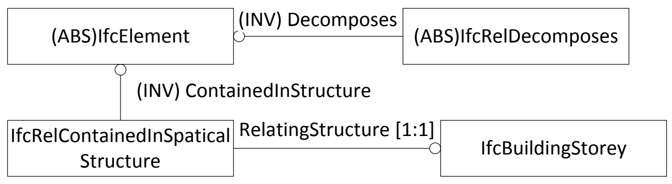
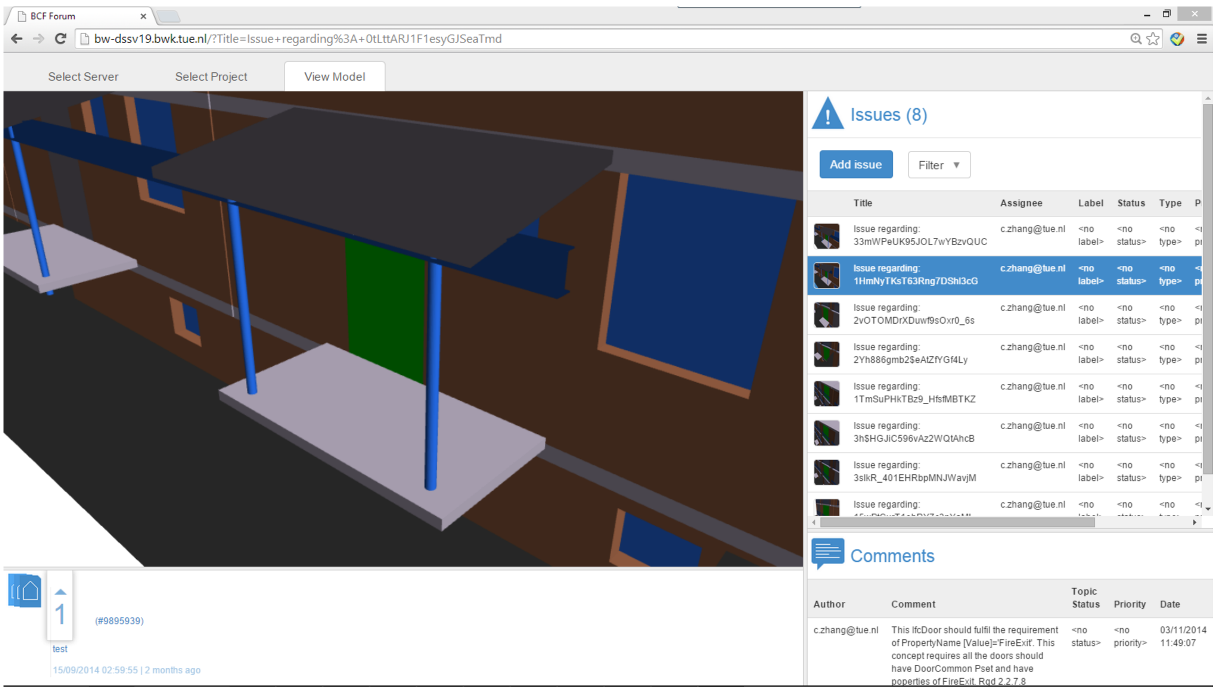


FIG. 6: Concept Template of the example rule in 4.4

5. CHECKING RESULTS

After check execution, BCF reports are generated. BCF reports and the checking tool are perfect partners and they benefit from each other. BCF reports make generated issues more visual and so more convenient for communication, while the checking tool automatically generates quantities of issues and their camera view point, which can fully take advantages of BCF files as opposed to manually generating BCF issues. However, this combination also raises questions about the management of generated issues and the integration of this check into working processes.

A BCF server is proposed and implemented by (Berlo and Krijnen 2014) to manage BCF issues generated from different parties. The "BCF forum" is an intermediary between bimserver.org and the BCF server. It provides a unified user interface to connect BCF issues in the BCF server with IFC model revisions in bimserver.org. It also supports zooming into specific parts of the building model according to the camera defined in BCF files. This model view checker has been added into this framework and semi-automatically works with them. According to the revision ids of IFC models on bimserver.org, their related BCF issues generated by the checker can be automatically associated with specific revisions of IFC models. Therefore, after checking for one revision of the IFC model, BCF reports are generated and submitted to the BCF server. Domain end-users can modify the design according to the issues presented on the BCF forum and submit the model again to the bimserver.org as another revision. This process recurrently continues until satisfaction of all rules. Fig. 7 shows a snapshot of the generated BCF reports opened in BCF forum. It shows the checking results of the rule defined in the sub-section 4.2.


BCF Forum
bw-dssv19.bwk.tue.nl/?Title=Issue+regarding%3A+0tLttARJ1F1esyGJSeaTmd
Select Server
Select Project
View Model
Issues (8)
Add issue
Filter
Comments
1
(#9895939)
test
15/09/2014 02:59:55 | 2 months ago

Title	Assignee	Label	Status	Type
Issue regarding: 33mWPeUK95JOL7wYBzvQUC	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 1HmNyTKsT63Rng7DShl3cG	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 2vOTOMDrXDuwf9sOxr0_6s	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 2Yh886gmb2$eAtZfYGf4Ly	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 1TmSuPHkTBz9_HfsfMBTKZ	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 3h$HGJiC596vAz2WQtAhcB	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding: 3slkR_401EHRbpMNJWavjM	c.zhang@tue.nl	<no label>	<no status>	<no type>
Issue regarding:	c.zhang@tue.nl	<no	<no	<no

Author	Comment	Topic Status	Priority	Date
c.zhang@tue.nl	This IfcDoor should fulfil the requirement of PropertyName [Value]='FireExit'. This concept requires all the doors should have DoorCommon Pset and have poperties of FireExit. Rgd 2.2.7.8	<no status>	<no priority>	03/11/2014 11:49:07

FIG. 7: Snapshot of a generated BCF files opened in BCF forum.

6. DISCUSSION AND LIMITATIONS

From this implementation, we found that although there are still some detailed implementation agreements that have to be made (e.g. quantifiers to deal with collection data types), the mvdXML standard and this implementation can be used as a light checking tool for IFC validation. However, there are still some general questions and limitations that are discussed in the following sections.

6.1 Reusability and efficiency

In the overall checking process, the time-consuming part is the structuring of model view rules. It is a long way from description-based requirements to low level selections of elements in the data model. There are many agreements and disambiguations that have to be made. For example, an external wall can be represented as an *IfcWall* with the "IsExternal" property set to "True", or has the *Name* "external wall" or has an associated *IfcClassification* with the *Name* "external wall", or has the *IfcPresentationLayerAssignement* "external wall" or a combination of all of the above. This process needs plenty of manual work and is error-prone. Therefore the improvement of the efficiency of this process and the reduction potential errors is an important question, which might be addressed e.g. through formalized implementers' agreements.

TABLE 3. Similar or overlapped rules between Rgd and Statsbygg.

Rgd BIM Norm	Statsbygg BIM Manual
§2.1.7 Model units, dimensions, display units, and rounding	9. Project units
§2.2.6.5 Geographic position and orientation	10. Defining and geo-referencing the project zero
§2.2.7.1 Project	11. Project, 33. Project
§2.2.7.2 Terrain	12. Site, 34. Site
§2.2.7.3 Building	13. Buildings
§2.2.7.4 Level	14. Storeys
§2.2.7.6 Space	15. Spaces-in general, 16 Spaces-functional, 22 Space-functional space heights, 36. Spaces
§2.2.7.5 Level Area object	18. Space-the gross area object
§2.2.7.7 Grouping of spaces:zone	26. Zones, 35. Functional zones
§2.2.7.8 Architectural, structural, and mechanical & electrical engineering elements	29. Modeling with both occurrence and type objects

In the General Requirements part of the Statsbygg BIM manual, there are many rules that are conceptually overlapping to the Rgd rules (Table 3). Therefore, there is a potential that developed rules can be reused across different standards to improve the efficiency. The built-in constructs *Concept Template* and *Concept* in mvdXML have improved the reusability of developed rules by allowing the assembly of new definitions from code snippets used earlier. For example, the rule defined in 4.1 as a *Concept* construct is required in both the Rgd and Statsbygg frameworks, thus it can be directly reused. We can also reuse developed *Concept Templates* to define new *Concept*s by changing parameters. For example, a rule like "a wall should have property IsExternal in the Pset_WallCommon" (Statsbygg 2011) can reuse the *Concept Template* defined in 4.2. We can apply this template to IfcWall and change the rule to "PropertyName[Value] = 'IsExternal' AND PropertySetName[Value] = 'Pset_WallCommon'". According to our experience, however there are still some factors obstructing reusing of developed rules. The first one is due to the current development environment of mvdXML which is restricted to very few tools and repositories. As the only open standard to define model views, mvdXML files are usually developed ad hoc with tools like ifcDOC (Chipman 2012). Very few reusable resources of model view concepts exist to date. Most of the existing ones are developed to limit the scope of the IFC schema to subsets but are not semantically strict enough to validate models for specific purposes. This fact however can be potentially solved by adding structured organization like shared repositories.

Another reason that may affect its reusability is that mvdXML is a semi-structured description method rather than a logic-based strictly formatted method. Developed *Concept Templates* and *Concepts* are identified by their names rather than actual semantics. Rules with the same semantics can be represented in multiple ways and different names, while different concepts can also be developed with the similar names and descriptions. For example, a cardinality restriction can be asserted in the *Cardinality* attribute or *Constraint* element in *Concept Template* level, and it can also be defined on *Concept* level. This might add more difficulties when considering the maintenance of already developed rules. Developers can reuse concepts and rules developed by themselves since they understand and remember their contents, but it is inconvenient for them to reuse resources from other parties. The constructs of *SubTemplates* and *SubConcepts* in mvdXML have provided possibilities to structure inheritance hierarchies between *Concept Templates* and *Concepts* to specify their semantics to some extent.

However, since mvdXML is not logic-based, currently these inheritance relationships are manually defined instead of automatically inferred. To address these issues, a more specific development guide might be needed for developers, meanwhile a formal and rigorous method is required to automatically structure concepts and rules and check their contents.

6.2 Ease-of-use

MvdXML is more easy-to-use than full-fledged programming languages and thus lowers the threshold for common-day use. However, it still requires development skills with a strong background in the IFC specification. The semantics of a *Concept* in mvdXML is more like a sentence of rule rather than a concrete concept in the real world. For example, a rule stating "every space except a service shaft should be accessible through at least one door" is defined as a *Concept* to specify this if-then dependency. However, from a domain expert's perspective, the concepts "service shaft", "is accessible through" and "door" might be more interpretable and (re-)usable. From the examples listed in section 4, we conclude that a mechanism to map low level elements in data models to human understandable terms can be developed. A Description Logic based method can be used to map elements from the IFC schema to natural language concepts. This approach requires the development of a collection of terminologies used for exchange requirements and rules.

6.3 Expressivity

The current version of mvdXML is mainly used to check the *explicit* information in IFC models. Therefore, to some extent it is a tool to check the agreements on information and their implementation rather than to check the semantics of models. For example, if we want to check whether the wall height complies with the building storey height (Rgd 2.2.7.8), the values of heights must be explicitly specified as properties. An inference mechanism based on their location and geometric properties to derive their *implicit* height properties and topological relationships to enrich and check IFC model is still missing. Many rules such as "internal and external doors should be modelled with the correct dimensions and placement" (Statsbygg 68) need spatial inference rules as the precondition for checking executions. This issue might be out of the range of an IFC validation, but it is important for a smoother collaboration process.

6.4 Issue fixing

Based on generated issue reports, domain end-users should be enabled to edit building models to add missing information, modify improper model constructs, or delete information that should not be asserted. However, when issues are caused by IFC implementations e.g. software vendors have disagreements with validation rules concerning how to capture information in IFC instances. These can hardly be solved in real working processes today. A simple example is that some applications such as Revit model "door type" in *IfcDoorStyle* instead of *IfcDoorType*, which however conflicts with many model views and BIM manuals. To identify implementation issues is one of the main purposes of IFC certification, but when these issues happen, domain users cannot directly fix them in projects. A rule language is a possible solution for this issue by transforming instances to different constructs. A similar discussion about this issue is provided in (Belsky et al. 2013).

6.5 Verification

This tool currently does not yet have mechanisms to evaluate if the checking result is completely reliable. Verification errors may of course occur due to implementation bugs, but verification errors caused by semantic errors within mvdXML rules happen more frequently. This again needs a formal method to review the contents of mvdXML rules to check if they correctly represent the meaning of the requirements or if developed rules have conflicts with each other. Transforming rules defined in mvdXML to logic-based method is a potential solution for this issue.

7. SUMMARY AND OUTLOOK

In this paper, a prototypical IFC validation tool based on open standards is presented. The tool and its source code are available on https://github.com/opensourceBIM/mvdXMLChecker. By developing rule-sets based on real-world BIM requirements, the mvdXML standard and this implementation have been introduced and tested. We also conceptually combine this checker with a BCF server to propose a possible workflow with this checker.

In addition, some general existing issues have been identified and discussed in the paper. We believe that these issues can potentially be addressed by a formal definition method based on logic theory.

With regards to future work, we are very interested in extending this work in two directions: The first one is to apply this tool and the proposed workflow in real or experimental projects in order to test them and get more feedback from users. Based on identified technical issues, the second perspective is to investigate the possibilities of logic theory based methods such as ontology and semantic web technology, which can be used as parallel methods or additional technical layers for the mvdXML standard to formalize developed model view concepts and rules. Preliminary research in this direction has been reported in (Zhang et al. 2014).

8. REFERENCES

Beetz J., Berlo L. van, Laat R. de and Helm P. van den. (2010). Bimserver.org - an open source IFC model server. *Proceedings of 27th International Conference on Applications of IT in the AEC Industry CIB-W78*, Cairo, November 2010. 1-8.

Belsky M., Sacks R. and Brilakis I. (2013). A framework for semantic enrichment of IFC building models. *Proceedings of 30th International Conference on Applications of IT in the AEC Industry CIB-W78*, Tsinghua University, Beijing, 514-523.

Berlo L. van, Beetz J., Bos P., Hendriks H. and Tongeren R. van. (2012). Collaborative engineering with IFC : new insights and technology. *Proceedings of 9th European Conference of Product and Process Modeling*, Reykjavik, Iceland, 811-818.

Berlo L. van and Krijnenb T.F. (2014). Using the BIM Collaboration Format in a server based workflow. *Proceedings of 12th International Conference on Design and Decision Support Systems in Architecture and Urban Planning*, Eindhoven University of Technology, Netherlands.

BIM Guides Project. (2014). http://bimguides.vtreem.com/bin/view/Main/WebHome.

Blis-project. (2009). IFC solutions factory – the Model View Definition site. http://www.blis-project.org/IAI-MVD/.

buildingSMART. (2011). Model View Definition summary, http://www.buildingsmart-tech.org/specifications/ifc-view-definition, accessed in January, 2014.

buildingSMART. (2013a). IFC 4 official release. http://www.buildingsmart-tech.org/ifc/IFC4/final/html/, accessed in January, 2014.

buildingSMART. (2013b). IFC toolboxes summary. http://www.buildingsmart-tech.org/implementation/get-started/ifc-toolboxes. Accessed in December, 2013.

Chipman T. (2012). ifcDoc Tool Summary. http://www.buildingsmart-tech.org/specifications/specification-tools/ifcdoc-tool/ifcdoc-beta-summary, accessed January 2013.

Chipman T., Liebich T. and Weise M. (2013). mvdXML: Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules. Model Support Group (MSG) of buildingSMART International Ltd..

Eastman C., Lee J., Jeong Y. and Lee J. (2009). Automatic rule-based checking of building designs, *Automation in Construction*, 18 (2009) 1011-1033.

Eastman C., Teichol P., Sacks R. and Liston, K. (2011). BIM handbook –a guide to building information modeling for owners, managers, designers, engineers, and contractors, 2nd edition, John Wiley & Sons Inc.

Hausknecht K., Liebich T., Weise M., Linhard K., Steinmann R., Geiger A., Hafele K.-H. (2014). BIM/IFC software certification process by buildingSMART, *eWork and eBusiness in Architecture, Engineering and Construction*, CRC Press, 129-133.

ISO 10303-11. (1997). Industrial automation systems and integration—Product data representation and exchange—: Description Methods: The EXPRESS Language Reference Manual, ISO Central Secretariat.

ISO 10303-14. (1999). Industrial automation systems and integration—Product data representation and exchange—: DescriptionMethods: The EXPRESS-X Language Reference Manual, ISO Central Secretariat.

ISO 16739. (2013). Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, ISO Central Secretariat.

IUG, International User Group of buildingSMART International Ltd. (2012). An integrated process for delivering IFC based data exchange. http://iug.buildingsmart.org/idms/methods-and-guides/Integrated_IDM-MVD_ProcessFormats_14.pdf/view, accessed December 2012.

Jeong Y.-S., Eastman C. M., Sacks R. and Kaner I. (2009). Benchmark tests for BIM data exchanges of precast concrete, *Automation in Construction*, 18, pp469-484.

Karlshoej J. (2012). Process and building information modelling in the construction industry by using information delivery manuals and model view definitions. *Proceedings of 9th European Conference on Product and Process Modeling*, 305–309.

Katranuschkov P, Weise M., Windisch R., Fuchs. S. and Scherer. R (2010) BIM-based Generation of Multi-Model Views. *Proceedings of. 27th International Conference on Applications of IT in the AEC Industry CIB-W78*, Cairo, 1-8.

NBIMS, National Institute of Building Sciences. (2007). National Building Information Model Standard Version 1.0-Part 1: Overview, Principles, and Methodologies. National Institute of Building Sciences.

Rillaer D. van, Burger J., Ploegmakers R. and Mitossi V., (2012). Rgd BIM Standard, version 1.0.1. 1–29. http://www.rijksvastgoedbedrijf.nl/english/documents/publication/2014/07/08/rgd-bim-standard-v1.0.1-en-v1.0_2.

Stangeland, B. K. (2011). BIM Collaboration Format. Available at: http://iug.buildingsmart.org/resources/abu-dhabi-iug-meeting/IDMC_017_1.pdf, accessed December 2013.

Statsbygg. (2011). Statsbygg Building Information Modelling Manual Version1.2. Available at: http://www.statsbygg.no/bim, accessed January 2014.

Lee G. (2011). What Information Can or Cannot Be Exchanged? *Journal of Computing in Engineering*, 25(1), 1–9.

Lee G., Park Y. H. and Ham, S. (2013). Extended Process to Product Modeling (xPPM) for integrated and seamless IDM and MVD development. *Advanced engineering informatics*, Vol. 27, Issue 4, 636 – 651.

OMG. (2011). Business Process Model and Notation version 2.0. Available on: http://www.omg.org/spec/BPMN/2.0/.

Parr T. (2007). The Definitive ANTLR Reference, Pragmatic Bookshelf.

NBIMS, National Institute of Building Sciences (2007). National building information model standard version 1.0-part 1: overview, principles, and methodologies, National Institute of Building Sciences.

Venugopal, M., (2011). Formal Specication of Industry Foundation Class Concepts using Engineering Ontologies. Ph.D. thesis, Georgia Institute of Technology, Atlanta.

Venugopal M., Eastman C., Sacks R., Teizer J. (2012). Semantics of model views for information exchanges using the industry foundation class schema. *Advanced engineering informatics*, 26 pp.411-428.

Weise M., Katranuschkov P. and Scherer. R. (2003) Generalized Model Subset Definition schema. In Construction IT: *Bridging the Distance, Proceedings of the CIB-W78 Workshop*.

Weise M. (2014). mvdXML requirements and examples: review of a standardized format to define and exchange model view definitions with exchange requirements and validation rules.

https://github.com/BuildingSMART/ mvdXML/tree/master/mvdXML1.1, accessed January 2014.

Windisch R., Katranuschkov. P. and Scherer. R (2012) A generic filter framework for consistent generation of BIM-based model views. *Proceeding of the International Workshop: Intelligent Computing in Engineering*, Technische Universität München.

Wix J. (2005). Information Delivery Manual: Guide to Components and Development Methods. Available at: http://iug.buildingsmart.org/idms/development/IDMC_004_1_2.pdf. , asccessed January 2014.

Yang D. and Eastman, C. M. (2007). A rule-based subset generation method for product data models. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 22, Issue 2, pages 133–148.

Zhang C., Beetz J. and de Vries B. (2014). An Ontological Approach for Semantic Validation of IFC Models. *Proceedings of the 21st International Workshop on Intelligent Computing in Engineering (EG-ICE)* 2014 - July 16th-18th, 2014 Cardiff, UK.