Model view checking: automated validation for IFC building models

C. Zhang
Eindhoven University of Technology, Eindhoven, Netherlands

J. Beetz
Eindhoven University of Technology, Eindhoven, Netherlands

M. Weise
AEC3 Deutschland GmbH, Germany

ABSTRACT: The validation for IFC building models is becoming increasingly important in BIM-based collaboration processes. In this paper we are reporting on a prototypical implementation of a model view checker for IFC validation. Although some checking methods have already been developed and implemented, they are based on proprietary methods which cannot be easily accessed, used and extended by end-users. This checker is developed based on the open mvdXML standard and is implemented on top of the bimserver.org framework. The checker is validated using rule-sets that were developed based on real-world BIM standards as test examples.

1 INTRODUCTION

In a collaboration environment like the building industry, being able to obtain information with sufficient quality is fundamental for working operations. Many researches have suggested that this objective can be approached by using vendor-neutral and open standards such as the Industry Foundation Classes (IFC) to capture and exchange data (Eastman et al 2011, Berlo et al 2012). As a general data model, IFC supports a full range of data exchanges among heterogeneous applications. It has provided a rich set of methods to capture data in order to be compatible with different domains and task scenarios. However, for particular tasks, there is very limited semantics in the underlying EXPRESS schema to specify which information should be exchanged and how it should be modelled (Venogopal et al. 2012). In the schema level "OPTIONAL" and weak typed attributes are the dominant citizens, and the model instances can also be semantically extended by mechanisms such as IfcPropertySet. All of these features have provided the flexibility required by different use-cases but also give too much freedom for application implementers and domain end-users.

Increasing specialization and automation in the building industry require high level interoperability. For many executions, 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). Therefore, on top of the IFC schema, additional rules and constraints should be specified to validate IFC building models to ensure that the receiving systems can reuse data and derive required information. Such validated IFC building models are the pre-condition for filtering data subsets and executing many automated tasks such as building performance analysis and code checking.

There are many related researches to address this issue. The standard methodology of the Information Delivery Manual (IDM) and Model View Definitions (MVD) has been introduced from the working process point of view to define required information for particular scenarios (Wix 2006, NBIMS 2007, IUG 2012). Various groups and consortia have developed IDMs and MVDs for their target domain model exchanges (Karlshoej, 2012). In addition to that, a number of national, company- and project-specific BIM standards and agreements have been developed such as the Dutch general service administration (Rijksgebouwendienst - Rgd) BIM Norm in The Netherlands (Rillaer et al. 2012) and Statsbygg BIM Manual in Norway (Statsbygg, 2011). These developed exchange requirements or BIM standards have defined description-based rules that IFC building models in specific context should conform to. In order to validate models, these requirements have to be converted to computer-executable rule-sets. Some commercial checking platforms or model servers such as Solibri and the EDMmodelServer have already implemented some of these technologies to check models. However, these requirements or rule-sets are implemented with proprietary methods, hard-wired in full-fledged high-level programming languages or other "black box" methods, thus they

cannot be easily accessed and flexibly reused by IFC experts and domain end-users. This is particularly problematic for SMEs that often do not have the resources to adapt these demanding technologies. It is inconvenient for these stakeholders to edit or develop rule-sets, to make new versions or define more detailed company- or project-specific rule-sets.

In this paper, we are reporting on a model view checker based on open standards to validate IFC building models. We show a prototypical implementation and test it with example use cases. The overall methodology and essential implementation details of this checker are introduced in the second section. In the third section, use-cases from two BIM standards are analysed. The last part provides a conclusion of the test use-cases, and discusses a future plan based on identified issues. 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 IMPLEMENTATION APPROACH

Unlike the rule checking for building designs, the aim of model view based checking is to validate model instances per se. Generally three steps are needed in this process: a) the interpretation of exchange requirements and structuring validation rule-sets, b) the check execution and c) report generation. This implementation is based on the open source bimserver.org framework (Beetz et al. 2009), 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.

2.1 *Development of model view rule-sets*

The mvdXML released by buildingSMART is an open standard to define model subsets and validation 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 "mandated", "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 ANTLR. It can be used to define more sophisticated rules either in "Concept Template" or "Concept" nodes to enhance the expressivity of the entire definition. For example, a rule like "every IfcWall should be typed by an IfcWallType" in IFC4 is defined as illustrated in Figure 1. 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 "mandated" to be applied on each IfcWall instance. On the "Concept" level, the type of value can also be specified further to say that the attribute of the RelatingType should be an IfcWallType (Fig. 1).

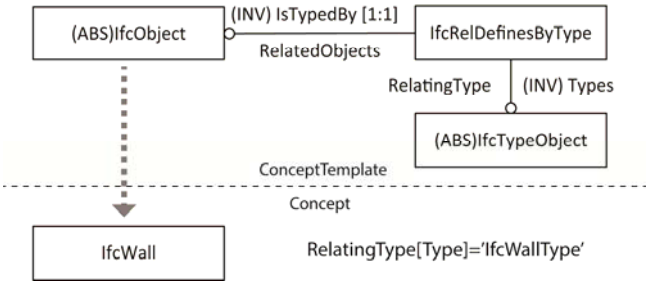

Figure 1. Example Model View Concept in mvdXML

2.2 *Check execution*

In an IFC instance file, the information is provided by the attributes of all existing objects. The bimserver.org platform already provides some mechanisms 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, 2009). 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 this case every IfcWall (including IfcWallStandardCase) will be checked if it has the related IfcRelDefinesByType which has the RelatingType of an IfcWallType. If one root object 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.

2.3 *Report generation*

After the checking execution, every generated issue will be captured in the form of a BCF report which includes a markup file and a viewpoint file (Stange-

land, 2011). 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 should define a camera based on the object's location and geometric attributes. For this communication purpose, we suggest that the root entity in mvdXML should be defined as a tangible entity (IfcProduct subtypes) if possible.

3 USE-CASES

In this research, use-cases from the Rgd BIM Norm and Statsbygg BIM Manual are used to define rulesets to test the checker. According to Weise (2014), the validation rules can be categorized as follows: a) checking data existence, including existence of attribute values, referenced entities and their cardinalities; b) checking content of values, including the value of simple data types and collection types; c) uniqueness of values; d) checking the conditional dependency (if-then) and consistency among them. The type a) usually accompanies with the later three, while d) is based on the checking results of a), b) and c).

3.1 *Rgd BIM Norm*

The Rgd BIM Norm in The Netherlands is a generic standard for BIM models. 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 calculation), a brief overview for all the rule types in the Rgd standard is listed in Table 1.

Table 1. Rule category of Rgd BIM Norm

Rule Types	Requirements in Rgd BIM Norm
a) data existence	§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
b)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
c)data uniqueness	§2.2.6.4, §2.2.7.6
d)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. For each type of rule, we give an example and provide two logic formulas using terms from description-based rules and IFC schema respectively to specify its semantics and make a comparison.

3.1.1 *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 should have a value or not and defines quantity requirements for the collection type attributes (Weise, 2014). 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).

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

When it is mapped to IFC Schema, this rule can be represented as (2).

$$\begin{aligned} \forall x(\text{IfcBuilding}(x) : & \exists z(\text{IfcRelAggregates}(z) \\ & \wedge \text{IsDecomposedBy}(x, z) \\ & \wedge \exists y(\text{IfcBuildingStorey}(y) \\ & \wedge \text{RelatedObjects}(z, y)))) \end{aligned} \quad (2)$$

In mvdXML, data existence and cardinality is defined by the cardinality rule in the "Concept Template", and it also can be further restricted by the token "[Size]" in the rules formatted by ANTLR grammar. This concept can be structured by the "Concept Template" of the basic aggregation relationship in the IFC schema (Fig. 2). 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'".

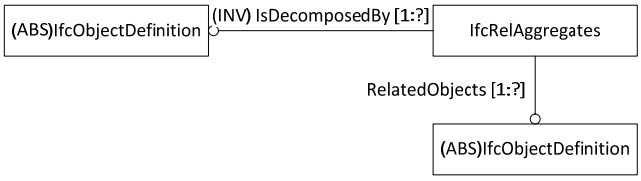

Figure 2. Concept Template of the example rule in 3.1.1

3.1.2 *Data content*

There are some common scenarios of data content rules in this standard. For examples, 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. For example, 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).

$$\forall x(Door(x) : \exists y(hasPropertySet(x, y) \wedge Pset\_DoorCommon(y) \wedge \exists z(containsProperty(y, z) \wedge FireExit(z)))) \quad (3)$$

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).

$$\forall x(IfcDoor(x) : \exists y(IsDefinedBy(x, y) \wedge IfcRelDefinesByProperties(y) \wedge (\exists z(RelatingPropertyDefinition(y, z) \wedge IfcPropertySet(z) \wedge Name(z, "Pset\_DoorCommon") \wedge \exists w(HasProperties(z, w) \wedge IfcPropertySingleValue(w) \wedge Name(w, "FireExit")))))) \quad (4)$$

In mvdXML, this type of rule is specially defined by the "[Value]" token. This example can be structured by the template as Figure 3. This template is applied on IfcDoor with additional rule of "PropertyName[Value] = 'FireExit' AND PropertySetName[Value] = 'Pset_DoorCommon'".

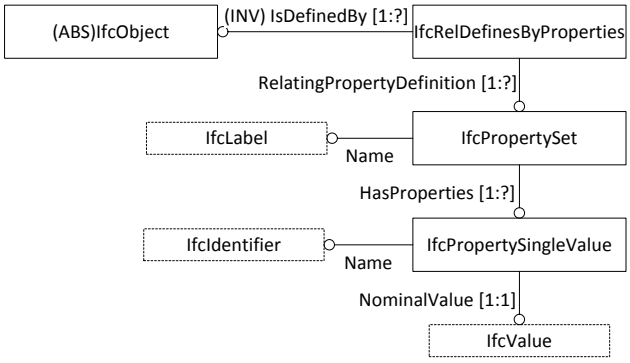

Figure 3. Concept Template of the example rule in 3.1.2

3.1.3 *Uniqueness*

The uniqueness checking rules are very few in this standard. Rgd §2.2.7.6 defines an implicit rule to specify that the space names should be unique. This can be represented as:

$$\forall x(Space(x) : \exists y(name(x, y) \wedge isUnique(y))) \quad (5)$$

This rule can be represented by the IFC elements as:

$$\forall x(IfcSpace(x) : \exists y(name(x, y)) \wedge \neg\exists z(IfcSpace(z) \wedge name(z, w) \wedge ((z \neq x) \wedge (y = w)))) \quad (6)$$

In mvdXML, a simple "Concept Template" can be structured (Fig. 4) and applied on IfcSpace. The attribute Name of IfcSpace can be defined as "Name[Unique] = TRUE".

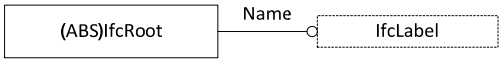

Figure 4. Concept Template of the example rule in 3.1.3

3.1.4 *Conditional rules*

The conditional rule is to check the dependency or consistency between object attributes. 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 does not compose other element, it should have association with a building level" specified in (7).

$$\forall x(Element(x) : \neg\exists y(composes(x, y)) \rightarrow \exists z(BuildingLevel(z) \wedge hasAssociation(x, z))) \quad (7)$$

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

$$\forall x(IfcElement(x) : \neg\exists y(Decomposes(x, y)) \rightarrow \exists w(ContainedInStructure(x, w) \wedge IfcRelContainedInSpatialStructure(w) \wedge \exists z(RelatingStructure(w, z) \wedge IfcBuildingStorey(z)))) \quad (8)$$

In the mvdXML standard, conditional dependency relationships are currently implemented by logic connectors. The conditional rules are defined as: "(Decomposes[Size] = 0 AND ContainedInStructure[Size] = 1) OR Decomposes [Size]=1".

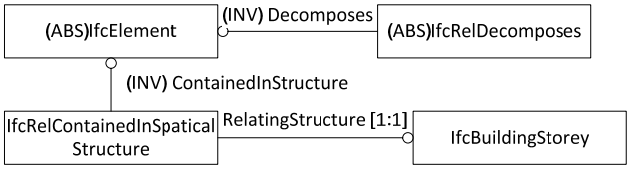

Figure 5. Concept Template of the example rule in 3.1.4

3.2 *Statsbygg BIM Manual*

The Statsbygg BIM Manual is a generic BIM guide in Norway. In its General Requirements part, there are many rules that are similar to the Rgd rules (Table 2). Therefore, there is a potential that developed Rgd rules can be reused for Statsbygg. Many "Concept Templates" can be directly reused to define the data existence type of rules, but there are still a number of disagreements for other rule types. The Statsbygg also has defined sets of domain specific requirements. The detailed examples are not presented in this paper.

Table 2. 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 georeferencing 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

4 RESULTS

4.1 *Report Generation*

After check execution, a set of BCF reports are generated. For example, Figure 6 is a snapshot of a generated BCF report opened in Solibri. This is one of the checking results of the rule defined in 3.1.2.

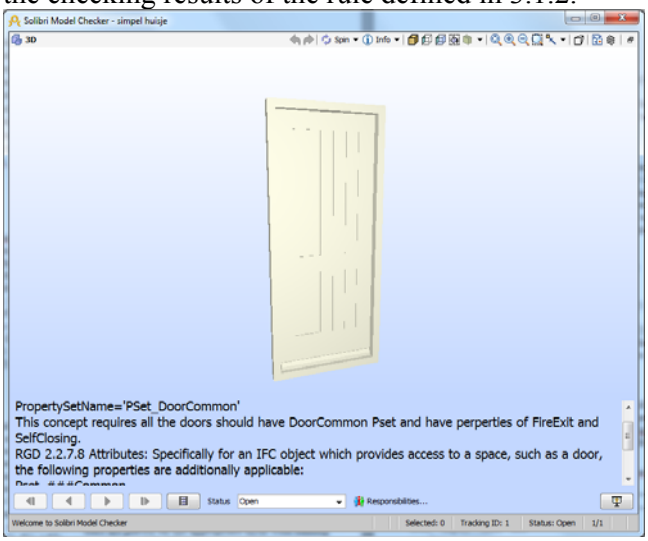

Figure 6. Snapshot of a generated BCF file opened in Solibri.

4.2 *Issues to be discussed*

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, we found that there are still some general questions worthwhile discussing.

4.2.1 *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 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 the IfcClassification with the Name of "external wall", or has the IfcPresentationLayerAssignement of "external wall" or all of the above. From Table 2, we can see that many rules are conceptually overlapping and so have the potential to be reused even across different standards. However, according to our experience, it is not easy to reuse developed concepts. This may due to different agreements mentioned before, but also due to the current development environment of mvdXML which is restricted to a 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). Although mvdXML has provided mechanisms like Concept Template and Concept to improve its reusability, 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. Another reason that may obstruct its reusability is that mvdXML is a semi-structured description method rather than a logic-based strictly formatted method. Rules with the same semantics can be represented in multiple ways, while different concepts can also be developed with the similar names and descriptions. It may add more difficulties when considering the maintenance of already developed rules.

4.2.2 *Easy-to-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 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 of "service shaft", "is accessible through" and "door" might be more interpretable and reusable. From the examples listed in 3.1, we believe 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.

4.2.3 *Data derive and inference*

The current version of mvdXML is mainly used to check the *explicit* information in IFC models. There-

fore, 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.

5 SUMMARY AND FUTURE WORK

In this paper, a prototypical IFC validation tool based on open standards is presented. By developing rule-sets based on real-world BIM requirements, the mvdXML standard and this implementation have been introduced and tested.
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 investigating the possibilities of logic theory based methods such as ontology and semantic web technology, which can be used as additional technical layers of the mvdXML standard for IFC validation.

REFERENCES

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

Berlo, L.van, Beetz, J., Bos, P., Hendriks, H., Tongeren, R. van 2012 Collaborative engineering with IFC : new insights and technology. In Proceedings of 9th European Conference of Product and Process Modeling.

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., 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., Lee.J. 2009. Automatic rule-based checking of building designs. Automation in Construction 18 (2009) 1011-1033.

Eastman, C., Teichol, P., Sacks, R., Liston, K. 2011. BIM Handbook –a guide to building information modeling for Owners, Managers, Designers, Engineers, and Contractors, 2nd edition. John Wiley & Sons Inc.

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.

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

Mazairac, W. & Beetz, J. 2012. Towards a framework for a domain specific open query language for building information models. In Proceeding of the International Workshop: Intelligent Computing in Engineering, München: Technische Universität München.

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., Mitossi, V., 2012. Rgd BIM Standard, version 1.0.1. 1–29.

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.

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

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

Wix, J. (2006). 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.