Technical University of Munich

Department of Civil, Geo and Environmental Engineering

Chair of Computational Modeling and Simulation

Checking mvdXML using mvdXML

Master Thesis

of the Master of Science degree in Civil Engineering

Author: Rawan Khaled Gaafar

Matriculation number:

1. Supervisor: Prof. Dr.-Ing. André Borrmann

2. Supervisor: Stefan Jaud, M.Sc

3. Supervisor: Sebastian Esser, M.Sc

Date of Issue: 02. May 2022

Date of Submission: 02. November 2022

Acknowledgment

Taking this opportunity, I would like to extend my deepest thanks and gratitude to my academic advisor M.Sc. Stefan Jaud for his positive guidance, understanding, and support over the past half year. I am grateful for his valuable expertise, counsel, inspiration, and motivation. My sincere thanks and deep gratitude go out also to Prof. Dr.-Ing. André Borrmann and M. Sc. Sebastian Esser for their helpful advice, guidance, and remarks during this work. Without your assistance and dedicated involvement in every step, this paper would have never been completed.

To my father, Khaled Gaafar, thank you for everything you do for us and for being the biggest support system for this family. To my mother, Yeldiz Helmy, thanks for always encouraging me to go forward; thank you for representing strength, sacrifice, determination, and unconditional love in my life. To my dear sister, you have changed and inspired me in many positive ways you may not even be aware of. Thank you for always being beside me, for being the symbol of power and strength to me, Sara. To my life partner, Omar Elsaqqa, thank you for all your love and support during this journey, emotionally, mentally, and academically.

Finally, I would also like to express my deepest gratitude to my friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

The digitization of the construction industry relies heavily on digital building models. Building Information Modeling (BIM) represents a built facility by creating models to store vast amounts of geometric and semantic attributes throughout its life cycle. Therefore, it is necessary to employ different software applications from various developers to enable a highly collaborative process among the involved parties to include all the aspects within a single model. The Industry Foundation Classes (IFC) standard was developed by buildingSMART International (bSI), enabling open, vendor-neutral data sharing for various use cases among heterogeneous applications. However, Software developers may find it challenging to support and transfer all this information, which may not be required in all cases. That is why Model View Definitions (MVDs) are established to reduce the scope of the facility information and to achieve automatic quality assurance through the specification of the Exchange Information Requirements (EIR). MVDs are currently encoded in the electronic format mvdXML. Even though mvdXML opens vast possibilities and has several use cases, such as documentation, filtering, and validation, there are presently several disputed uncertainties about the robustness of mvdXML. Moreover, despite the many scientific research studies based on mvdXML in validation procedures of IFC, up to now, there is no validation technique to check the consistency of mvdXML itself.

This thesis provides a prototypical implementation of validating mvdXML using mvdXML. A checker based on the mvdXML as the format for structuring the validation rules to check other mvdXML files is developed. In this paper, two main aims are presented: 1) to develop a standardized validation tool for mvdXML; 2) to examine how robust mvdXML is and how versatile mvdXML can be by challenging its capabilities and taking it to a new meta-level to be used to check itself instead of only using it to check IFC files. The method is verified using three different datasets, and the main problems, difficulties, and requirements are discussed. Ultimately, we came up with a successful methodology to check mvdXML files using mvdXML. This encourages other users to create their mvdXML files confidently since there is now a method to check those files. The methodology can also be further developed for any data model since it has already been successfully used on itself, proving its robustness and versatility

abilities. However, some limitations exist within the scope of mvdXML 1.1 for defining the checking rules. Hence, in future versions of mvdXML, it might be of interest to expand the schema to support more features to eliminate those limitations.

Zusammenfassung

Die Digitalisierung der Bauindustrie stützt sich in hohem Maße auf digitale Gebäudemodelle. Die Gebäudedatenmodellierung (Building Information Modeling, BIM) stellt eine gebaute Einrichtung dar, indem sie Modelle erstellt, in denen große Mengen geometrischer und semantischer Attribute während ihres gesamten Lebenszyklus gespeichert werden. Daher müssen verschiedene Softwareanwendungen von unterschiedlichen Entwicklern eingesetzt werden, um einen hochgradig kollaborativen Prozess zwischen den beteiligten Parteien zu ermöglichen, damit alle Aspekte in einem einzigen Modell berücksichtigt werden können. Der Industry Foundation Classes (IFC)-Standard wurde von buildingSMART International (bSI) entwickelt und ermöglicht den offenen, ortsneutralen Datenaustausch für verschiedene Anwendungsfälle zwischen heterogenen Anwendungen. Für Softwareentwickler kann es jedoch eine Herausforderung sein, all diese Informationen zu unterstützen und zu übertragen, was nicht in allen Fällen erforderlich ist. Aus diesem Grund wurde die Model View Definition (MVD) eingeführt, um den Umfang der Anlageninformationen zu reduzieren und eine automatische Qualitätssicherung durch die Spezifikation der Exchange Information Requirements (EIR) zu erreichen. MVDs werden derzeit in dem elektronischen Format mvdXML kodiert. Auch wenn mvdXML weitreichende Möglichkeiten eröffnet und verschiedene Anwendungsfälle wie Dokumentation, Filterung und Validierung bietet, gibt es derzeit einige umstrittene Unsicherheiten hinsichtlich der Robustheit von mvdXML. Darüber hinaus gibt es trotz zahlreicher wissenschaftlicher Untersuchungen, die mvdXML in Validierungsverfahren der IFC einsetzen, bisher keine Validierungstechnik, um die Konsistenz von mvdXML selbst zu überprüfen.

Diese Arbeit bietet eine prototypische Implementierung der Validierung von mvdXML mit mvdXML. Es wird ein Checker entwickelt, der auf mvdXML als Format zur Strukturierung der Validierungsregeln basiert, um andere mvdXML-Dateien zu überprüfen. In diesem Beitrag werden zwei Hauptziele vorgestellt: 1) die Entwicklung eines standardisierten Validierungswerkzeugs für mvdXML; 2) die Untersuchung der Robustheit und Vielseitigkeit von mvdXML, indem die Fähigkeiten von mvdXML herausgefordert werden und es auf eine neue Metaebene gebracht wird, um sich selbst zu prüfen, anstatt es nur zur Prüfung von IFC-Dateien zu verwenden. Die Methode wird anhand von drei

verschiedenen Datensätzen verifiziert, und die Hauptprobleme, Schwierigkeiten und Anforderungen werden diskutiert. Letztendlich haben wir eine erfolgreiche Methode zur Prüfung von mvdXML-Dateien mit mvdXML entwickelt. Dies ermutigt andere Benutzer, ihre mvdXML-Dateien mit Zuversicht zu erstellen, da es nun eine Methode zur Überprüfung dieser Dateien gibt. Die Methodik kann auch für jedes andere Datenmodell weiterentwickelt werden, da sie bereits erfolgreich für sich selbst verwendet wurde, was ihre Robustheit und Vielseitigkeit unter Beweis stellt. Allerdings gibt es im Rahmen von mvdXML 1.1 einige Einschränkungen bei der Definition der Prüfregeln. Daher könnte es in zukünftigen Versionen von mvdXML von Interesse sein, das Schema zu erweitern, um mehr Funktionen zu unterstützen und diese Beschränkungen zu beseitigen.

Contents

List of Figures

List of Tables

List of Abbreviations

AEC	Architecture, Engineering, and Construction
DSRM	Design Science Research Methodology
BIM	Building Information Modelling
IFC	Industry Foundation Classes
BCF	BIM Collaboration Format
bSI	BuildingSMART International
MVD	Model View Definitions
UML	Unified Modeling Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
OOM	Object Oriented Modeling
SGML	Standard Generalized Markup Language
W3C	World Wide Web Consortium
URI	Uniform Resource Identifier
OOP	Object Oriented Programming
IDM	Information Delivery Manual
LOI	Level of Information
DTD	Document Type Definition
XSLT	eXtensible Stylesheet Language Transformations

EIR	Exchange Information Requirements
ER	Exchange Requirements
ERM	Exchange Requirements Model
IR	Information Requirements
PDT	Product Data Template
DD	Data Dictionary
IDS	Information Delivery Specification
DOC	Document
XLS	Excel Spreadsheet File
PDF	Portable Document Format
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
STEP	Standard for the Exchange of Product
IfcDoc	IFC Documentation
HTML	HyperText Markup Language
ifcQL	IFC Query Language
RDF	Resource Description Framework
OWL	Web Ontology Language
PAT	Process Analysis Toolkit
KIT	Karlsruhe Institute of Technology
UUID	Universally Unique Identifier
COBie	Construction-Operations Building Information Exchange

1 Introduction

According to international standards, Building Information Model (BIM) is "A shared digital representation of physical and functional characteristics of any built object including buildings, bridges, roads, process plants, etc., forming a reliable basis for decisions" (ISO 29481-1:2010). BIM has become an essential compatible method, supported by various tools and technologies, constantly used in the construction sector to facilitate communication and collaboration among project participants. BIM-based on open standards aims to include every aspect of the building industry. The Industry Foundation Classes (IFC) is an open and standardized data model with the goal of promoting software application interoperability for building information modeling in the Architecture, Engineering & Construction industry (AEC). (Laakso & Kiviniemi, 2012, p. 135)

Since BIM is used to digitally represent a facility's geometric and semantic attributes throughout its life cycle, the quantity of data could be huge. Most of the time, it is not easy to transfer all of the model's information because it may be far more extensive than what is necessary for a certain task. That is why Model View Definitions (MVD) were developed primarily for IFC implementation and have been viewed as subsets of the IFC Model Specification (Hietanen & Final, 2006, p. 2). MVDs can be encoded in an electronic format called mvdXML, mainly used to aid the filtering and validating of IFC instance files. Although mvdXML is a data model itself, it currently does not yet have any mechanisms for checking it.

1.1 Motivation

This paper aims to validate mvdXML using itself. The motivation stemmed from the fact that mvdXMLs can check IFC, but what can validate mvdXML files themselves? That raised the research question of whether we can check mvdXML using mvdXML since it can already check IFC instances. Furthermore, another goal of this work, which is the main reason why we have chosen mvdXML, is to analyze the capabilities of mvdXML to check a data model other than IFC. We want to examine how mvdXML can be used to check another data model outside IFC's scope, such as mvdXML itself.

This work implements a prototypical validation technique for quality control of mvdXML using itself. Thus, several different checks are created to not only validate many

aspects of an mvdXML file, but also to push mvdXML as a validation tool to its limits and challenge it with as many different combinations of check types as possible. Not only the correctness of any mvdXML file is guaranteed by using this method, but also significant findings about using mvdXML within another scope other than IFC can be reached, which in return help in defining new requirements for the subsequent versions.

1.2 Structure

In the scope of this thesis, the Design Science Research Methodology (DSRM) is employed, as established by Peffers et al. (2007). DSRM includes six main steps (Peffers et al., 2007): -

1. Problem identification and motivation
2. Definition of the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication.

This paper consists of 7 main chapters: introduction, theoretical background, related work, methodology and implementation, testing and results, and discussion. The introduction describes the motivation, the proposed solution, and its objectives. The theoretical background briefly discusses an overview of the scientific literature and some theory concepts required for this work. In chapter 3, some of the prior works that influenced this work are examined. Chapter 4 consists of the third step in the DSRM. It explains the proposed validation method, including the error scenarios, and checks definitions. Also, this part contains insight into the different databases used to test this tool, including the designed furniture example. Chapter 5 describes the demonstration, including the experiments and findings. It displays the results of all the experiments conducted for this study using graphs, charts, and comparisons. In chapter 6, detailed justifications behind the findings are described, along with a general analysis of the capabilities and limitations of mvdXML. Hence, the whole method is evaluated. Chapter 7 is the last part of this work, which concludes the whole work and discusses some of the future work. The crucial points covered throughout the thesis are summarized, along with the study's constraints and potential directions for further research.

2 Theoretical Background

This chapter discusses some definitions and examples of the main topics used in this work. It introduces some generic terms, then gets more profound into the main topic.

2.1 Data Modeling

Data modeling in the AEC industry consists of two main features: geometry and semantics. In most cases, when digitally modeling building and infrastructure systems, it is essential to consider the geometric and semantic data. To support the design, planning, construction, and operation of a physical facility, information can be digitally gathered, structured, analyzed, summarized, compared, and appraised with the help of digital building models. In general, digital and prototypical models come in various forms regarding different disciplines and fields. These all attempt to provide a digital copy of reality or to reflect an as-yet-unrealized aspect of a planned future reality digitally. One significant form in the AEC industry is Building Information Models. (Borrmann et al., 2015, p. 44)

This chapter covers the essential terms and concepts related to data modeling that will be covered in later chapters. Since, as known and mentioned, IFC and mvdXML are data models, the most important terms and concepts for data modeling, including data modeling languages, schema, and instances, as well as classes and objects, are introduced.

2.1.1 Data Modeling Languages

Numerous languages can define and describe the data models. The Unified Modeling Language (UML) and the Extensible Markup Language (XML) are introduced in this section since they are used in the subsequent chapters within the thesis's scope.

2.1.1.1 Unified Modeling Language (UML)

The Unified Modeling Language (UML), a language that has been internationally defined and standardized, is used to graphically represent object-oriented models (OOMs) in a variety of diagrams or aspects using texts and symbols (Booch, 2005). Class and Object diagrams are the most important types of UML diagrams. Class

diagrams illustrate the principles of a class regarding its type, attributes, methods, and relationships to the other classes in structured representations of a particular domain (Borrmann et al., 2018, p. 46). A class is depicted on the class diagram along with three horizontal sectional divisions, as shown in Figure 2.1. The upper section describes the class's name, the middle section includes the class's variables, and the below section includes the class's functions or methods. This class can be joined together with other classes to form a complete class diagram illustrating their relationships. In the same way, an object diagram is a class diagram that shows the detailed state of a class at a point in time.

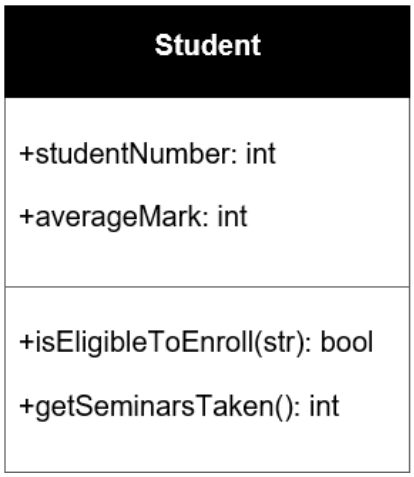

Figure 2.1: Class diagram with three compartments (www.diagrams.net/blog/uml-class-diagrams)

2.1.1.2 eXtensible Markup Language (XML)

The World Wide Web Consortium has standardized XML, or Extensible Markup Language, as a structured markup language to store and transmit text documents. XML is a straightforward and highly adaptable text format derived from SGML (ISO 8879:1986), the Standard Generalized Markup Language. XML was initially created to address the difficulties of large-scale electronic publishing. However, it is becoming more crucial in transmitting a range of data on the Web and elsewhere. (W3C, www.w3.org)

Serialization is the primary function of XML. It keeps information in a very structured way, and XML documents are readable by computers and humans. XML tags, referred to as "markup," represent the data structure. An element can be expressed in XML by writing its name between opening tags and ending it again with its name between closing tags. Between the opening and closing tags are the data and attributes describing the elements (for example, <Door>……</Door>). (Borrmann et al., 2018, p. 47)

Additionally, namespaces are used in an XML document to provide elements and attributes with unique, distinctive names. Names for elements or attributes in an XML instance may come from different XML vocabularies. The ambiguity between identically named elements or attributes can be eliminated if each vocabulary is given its namespace. By connecting element and attribute names used in XML documents with namespaces indicated by URI references, XML namespaces offer a quick way to qualify the names used. The reserved XML attribute xmlns or xmlns:prefix, whose value must be a valid namespace name, is used to declare an XML namespace. (W3C, www.w3.org)

Defining a data model (XML schema file) and storing the actual data (XML data file) are both possible using XML. An XML schema defines the structure of an XML document that goes beyond the fundamental syntactical restrictions imposed by XML itself. The additional XML schema, usually XML Schema Definition (XSD), defines the required data for parsing and validating XML. (Dykes & Tittel, 2011)

Since one of the formats used to store schemas and instance data is XML, and one of the most common descriptive languages used for defining a schema is XSD, several document formats make use of it (Borrmann et al., 2018, p. 222). MvdXML is one of those document formats, which uses XML to serialize the information, and is based on the XSD.

2.1.2 Classes and Objects

In Object Oriented Programming (OOP), a class, such as the "Chair" class shown in Figure 2.2, is a template that creates objects which share common variables and methods. It represents a template for creating instances, which are also called objects. An object's state is stored in variables, and its behavior is exposed through methods. (Borrmann et al., 2018, p. 48)

In other words, an object is a member of a specific class with predetermined values instead of variables. Instantiation can therefore be referred to as construction. Every object has a unique copy of variables that are not shared with other objects. In OOP, a special method in the class, a constructor, is called to create an object. (Wu, 2006)

Figure 2.2 illustrates the “Chair” class, while Figure 2.3 visualizes an object, Chair4, constructed from the chair class template.

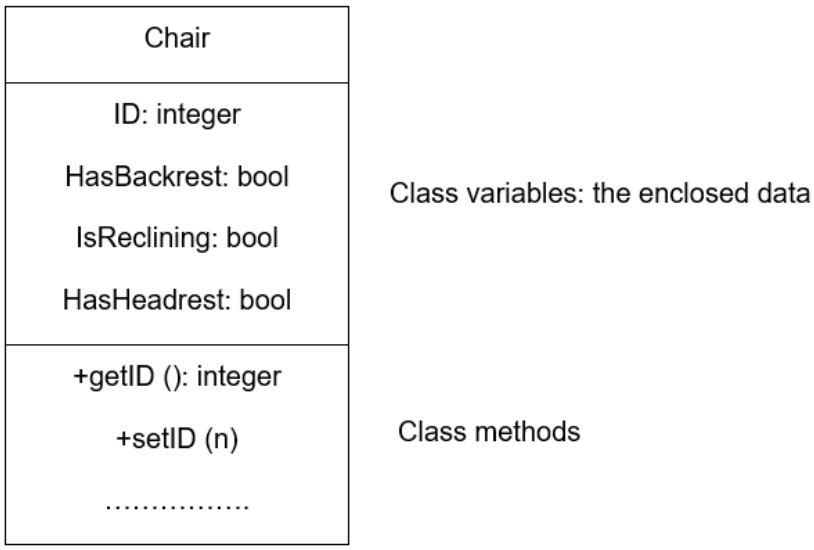

Figure 2.2: Declaration of class attributes and methods

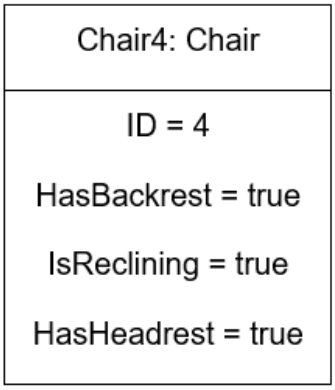

Figure 2.3: Illustration of object diagram of a chair class instance

2.1.3 Schema and Datasets

Cerovsek (2011) defines a schema as a standardized meta-model that specifies and defines the structure of schema-derived instance files. It identifies the object's characteristics, types, and restrictions (Cerovsek, 2011, p. 226). In other words, the outline for the data sets' construction is defined by a schema, while an instance of the database is the collection of data stored at any given time.

The following schema UML diagram in Figure 2.4 represents an example. It illustrates a schema that is composed of a collection of relatable classes. Each class is connected to other classes by a connecting line representing their relationship. The example below shows that the "Chair," "Armrest," and "Leg" classes inherit from the more generic class "Component." The arrow with a blank head serves to represent inheritance. Moreover, the "Chair" class is composed of "Armrest," "Leg," and "Production" classes. Cardinality is expressed near the end of the arrows, showing the required number of

instances from this class. A relationship's optionality can be either "0..", indicating that it is optional, or "1..", meaning it is required. At the same time, the cardinality can be specified from "..1" to "..*", indicating that only one instance or an unlimited number of class instances is required.

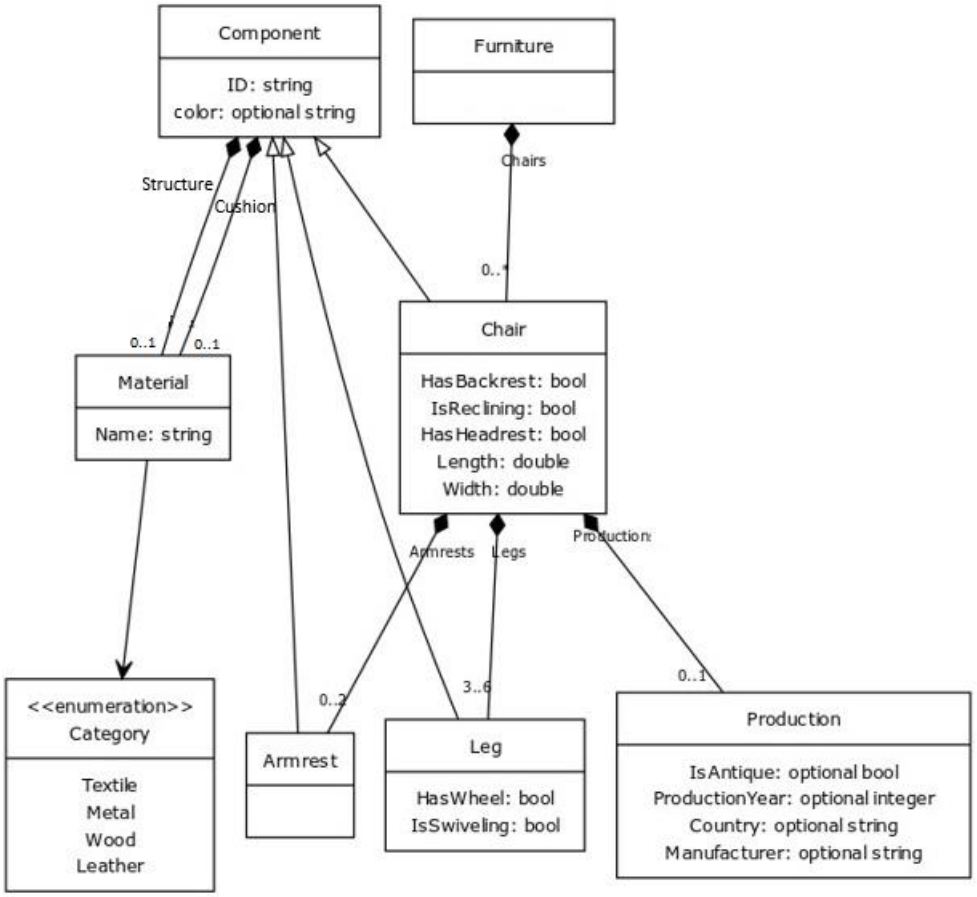

Figure 2.4: UML class diagram of furniture schema

2.2 Building Information Modeling (BIM)

BIM is the process of creating a precise virtual representation of a building asset. This data-rich three-dimensional geometric model can be utilized throughout the whole building life cycle. (Azhar, 2011, p. 241)

OpenBIM is a multidisciplinary area in which the modeling process is software independent. It is widespread in OpenBIM to link diverse models developed by different software programs. To enable interoperability between the different models, the Industry Foundation Classes (IFC) have been developed as an open standard in the OpenBIM area to serve the BIM interoperability necessities. (Laakso & Kiviniemi, 2012, p. 134)

The broad scope of application of OpenBIM prompts a massive quantity of data, which could sometimes be challenging to support in only one data model and to transfer all the information, which may not be required in all cases. Therefore, Model View Definition (MVD) was established by buildingSMART to control the scope of the IFC schema and assure the quality of the model information (Hietanen & Final, 2006). In addition, buildingSMART also established the standard mvdXML to represent MVDs and their supporting exchange requirements (ER) in an electronic format as documentation. Besides, mvdXML can be used to validate whether the data content of an IFC file written against an MVD complies with these requirements or not. (Chipman et al., 2012, p. 11)

This chapter introduces the topic, laying out the fundamentals of MVD and mvdXML so that the subsequent chapters can build on these principles.

2.2.1 Exchange Information Requirements (EIR)

Throughout the design, construction, and operation phases, building information models must be shared and exchanged according to unique requirements for each domain industry. EIR is a document specified by the client that outlines all the requirements pertaining to the information exchanges in a BIM process. The appointed party must meet these requirements during the handover (Borrmann et al., 2015, p. 240). Building model data transferred between building project professionals are governed by Model View Definitions (MVDs), which specify the information that must be shared. The information identifies model semantics that two or more applications must share. (Lee et al., 2016)

2.2.2 Industry Foundation Classes (IFC)

Industries need a neutral file format that includes their specifications and requirements for a variety of exchanges that support the full use of a building model. The buildingSMART international organization has produced the IFC file format to serve as a uniform platform for software developers. IFC schema is an open and standardized data model designed to allow BIM software programs in the Architecture, Engineering, and Construction (AEC) industry to communicate with one another (Laakso & Kiviniemi, 2012, p. 134). It is now commonly used not only to transfer data between different parties involved in the AEC industry, but also to archive project information throughout the different phases of the project. IFC data can be encoded in various formats, including XML, JavaScript Object Notation (JSON), and Standard for the

Exchange of Product (STEP). (buildingSMART, www.technical.buildingsmart.org/standards/ifc/ifc-formats)

In most circumstances, transferring all of the information that is included in the model is challenging, and this information could be much more than what is required for a specific task. Hence, Model View Definitions have been thought of as subsets of the IFC Model Specification and were created mainly for IFC implementation (Hietanen & Final, 2006, p. 2).

2.2.3 Information Delivery Manuals (IDM) and Model View Definitions (MVD)

In the previous section, IFC was introduced as a data model that can carry data related to a facility model throughout its life cycle. Because all information may be joined into a single model, this model can be highly complicated. To address this issue, buildingSMART has established the minimalist standardization approach to control which information is required for a particular use at which time (Borrmann et al., 2015, pp. 130 – 133). MVDs and IDMs are used to implement the approach.

IDM is a documented method for outlining the information exchange requirements. The fundamental goal of IDM is to ensure that specific important information and requirements are communicated at a particular time throughout the model life cycle. The end user's value chain specifies these requirements. An MVD is then used to guarantee that IFC implementations meet those needs. The main purpose of MVD is to constrain the scope of the entire IFC schema by specifying a subset of the IFC schema that is required and sufficient to meet one or more specific sets of data exchange requirements defined by the user in the IDM. Besides, they can be used for a wide range of other purposes, including automatic data quality assurance and certification of an IFC model. Depending on the range of its domain, a software application can execute more than one MVD. (Hietanen & Final, 2006, pp. 2-8)

An MVD is created from an IDM utilizing a portion of the IFC model. An MVD comprises predefined concept templates and specifications that can be reused. A concept uses a relational structure and a constraint to represent the necessary entities, attributes, properties, and relationships (Lee et al., 2016, p. 355). MVDs were very closely tied to the complete schema up until IFC 2x3. In IFC4, there are only two official MVDs: Reference View and Design Transfer View (BuildingSMART, https://technical.buildingsmart.org/standards/ifc/mvd/). The Reference View is designed to bring partial and main models from different authors together and smooth the coordination between

them, while The Design Transfer View represents the transfer of the entire model and any further modifications (Borrmann et al., 2015, p. 134). There are already another five MVDs in progress. Furthermore, the present direction is also going towards defining more MVDs due to the expansion of IFC into more fields, such as roads and bridges. (BuildingSMART, https://technical.buildingsmart.org/standards/ifc/mvd/)

2.3 State of the art of mvdXML

MVDs and accompanying exchange requirements are encoded in the electronic format mvdXML, designed by buildingSMART (Chipman et al., 2016, p. 7). MvdXML is the XML format that includes MVD information (Jiang et al., 2019, p. 4). This section examines the structure and purpose of mvdXML in greater depth.

The Model View Definition standard approach determines the rules that must be followed and the data exchange requirements for building information models. MvdXML is the formal machine-readable representation of MVD, recommended by BuildingSMART (Liu et al., 2019, p. 12). Thus, the main goal of why mvdXML was developed is to choose and specify the necessary information entities from a schema, their properties, and rules for specific exchange use cases. Later, the focus on using mvdXML as a validation tool was also considered. (Weise et al., 2017, p. 4)

The process of mvdXML development follows mainly four primary steps: 1) IDM definition, 2) MVD concepts definition, 3) application deployment and certification, and 4) BIM validation. Domain experts first establish IDMs to offer human-readable definitions for use-case scopes and EIRs of specific exchange cases. The IDM is then used to extract model subsets “model views” from IFC. These model views are usually encoded in mvdXML and can then be implemented in domain applications to develop export or import procedures for different exchange scenarios. In other words, IFC-based model views are subsets of the IFC schema tailored to end-user requirements. Instance models imported and exported from software programs are checked in the final BIM validation step to ensure the given ERs are met. (Zhang et al., 2013, p. 2)

Many professional groups have created IDMs and MVDs for targeted domain model exchanges using this methodology. A list of ongoing projects is available on the website of buildingSMART-tech.org. Phases 1 and 4 of the development process primarily focus on end users. However, phases 2 and 3 are more critical from a technical standpoint for the deployment of IFC because they are expected to embody the semantics captured and stated in ERs completely. As a result, throughout this procedure, it is

necessary to provide a set of information describing the entities that should be chosen, the features that should be mandatory or optional, and the desired level of detail. Since the IFC model is generic with many optional characteristics and offers a variety of techniques to identify objects and relationships, it makes this task complex and inconsistent. In order to map the exchange requirements to the IFC schema, a logical, formal procedure is required (Zhang et al., 2013, p. 2).

2.3.1 MvdXML Definition Tools

BuildingSMART has established the mvdXML as the official defined specification data format for storing MVDs, and it is not dependent on any specific product model schema. Any text editor can create a Model View Definition based on mvdXML. However, it is envisaged that specialized software will be utilized to read and write mvdXML data sets. Some of the specialized software that may be used to work with mvdXML is reviewed in this work.

The IFC Documentation Generator (ifcDoc) is a program used to create and edit mvdXML. It offers users a graphical user interface via which they can define all of the material within mvdXML, such as adding entities, characteristics, and constraints to concepts based on established exchange requirements. Additionally, this program can automatically construct instantiation diagrams, produce Hypertext Markup Language (HTML) documentation for model views, and create IFC4 documentation. MvdXML can also be edited in its raw form using XML/XSD editors like Microsoft Visual Studio and Eclipse, just like any other XSD-based schema. Moreover, MvdXML can be read by IFC-based software programs to filter and validate data automatically so that it complies with the given restrictions. Additionally, mvdXML can be written by IFC-based software applications to let users define unique exchange circumstances (Chipman et al., 2016, p.8). We use Microsoft Visual Studio within this thesis's scope for developing the mvdXML checks.

2.3.2 MvdXML Usage

MvdXML serves a number of purposes. The primary purpose of mvdXML is to customize and confine the entire IFC model to specific subsets which are sufficient for certain use cases. MvdXML supports documenting the model views, as well as aid in the filtering and validating of IFC instance files. A subset of mvdXML could be sufficient to

provide the necessary functionality in some instances requiring only one purpose. Although mvdXML is utilized mainly with IFC, it is not limited to IFC since it is a generic framework and can be used with completely different schemas. (Chipman et al., 2016, p.7)

2.3.3 MvdXML Schema Structure

MvdXML schema, shown in Figure 2.5, comprises two main sub-components: mvd:Templates and mvd:Views. The single root element mvdXML defines the two sub-components. The definition of reusable concept templates, mvd:ConceptTemplate, is at the heart of the mvd:Templates function. Mvd:ConceptTemplate specifies sub-graphs that contain all schema information required for the functional unit addressed by the concept. Views have a collection of model views, mvd:ModelView, which clarifies how concept templates should be applied to fulfill the exchange requirements. (Chipman et al., 2016, p. 10)

MvdXML rules are separated into two parts: data structures and rule statements. IFC data structures are described as nested XML tags in the header, which indicate how to find related data nodes starting from a root entity nodeset by nested XML tags. (Liu et al., 2019, p. 12). The concept template is the part that encompasses the data structures, beginning with an applicable root entity, which is usually a subtype of IfcObject. The root entity continues with a list of AttributeRules and EntityRules repeated successively, right down to the required attributes to define the unit of functionality. (Chipman et al., 2016, pp. 10-11).

The mvd:ModelView contains information on how concept templates are utilized in a view to define mvd:ExchangeRequirements. The exchange requirements element is one of the significant elements of the model view. It specifies whether or not the mvd:TemplateRules in each mvd:ConceptRoot must be met in a specific use case. The second main part of the model view is mvd:Roots, which is comprised of a set of mvd:ConceptRoots. Concept root defines mvd:Concepts as well as template rules that apply to a specific IFC item by referring to it. Mvd:TemplateRules define the rules which a concept needs to pass for validation. Additional limitations could be imposed using the mvd:Applicability tag. If applied, restrictions must be met by the entity instance before applying the concepts. (Chipman et al., 2016, pp. 10-13)

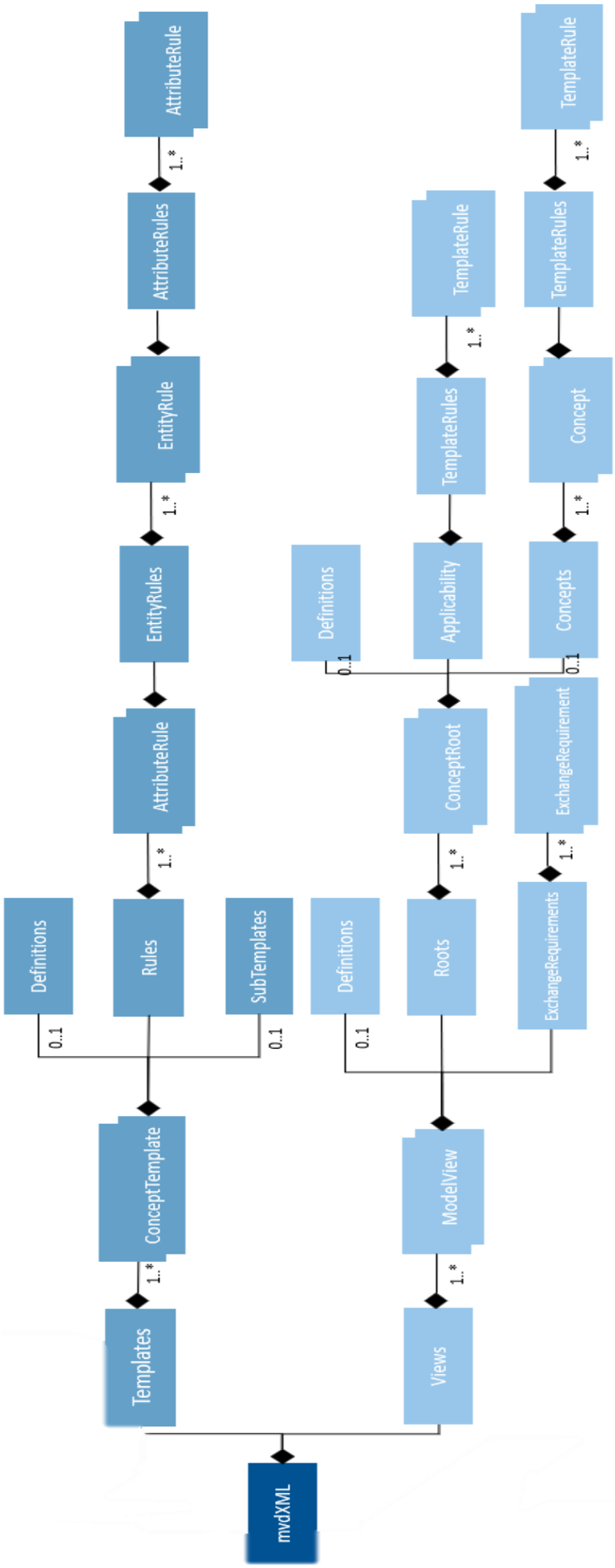

Figure 2.5: The mvdXML schema's basic structure (Chipman et al., 2016)

2.3.3.1 MvdXML Schema Components

An mvdXML file starts with the single root element, mvdXML, which includes zero-to-many mvd:ModelView and zero-to-many mvd:ConceptTemplate. The number of concept templates should be, at the very least, the same as the number referred to in the provided model view(s). (Chipman et al., 2016, p. 18)

ConceptTemplate element may also have mvd:SubTemplates and can create a tree of connected reusable concept templates. It might refer to related partial concepts within the tree. The following are the relevant optional and required attributes and elements that a concept template element have. (Chipman et al., 2016, pp. 18-19):-

- @applicableSchema: provides information on the default schema to which the template applies, such as IFC2X FINAL, IFC2X2 FINAL, IFC2X3, or IFC4.
- @applicableEntity: an array of strings that specify the IfcRoot-based entities that the concept applies to, including all derived entities. If a subTemplate is used, the applicable entity must be the same type or subtype of the main template.
- Rules: a collection of attributes (AttributeRule) declared at the applicableEntity, where each attribute may have defined graphs of object instances or value constraints.

The "Rules" element type is AttributeRule, i.e., it has a collection of AttributeRule elements. The AttributeRule element specifies an attribute for an entity together with any associated restrictions and/or entity rules. The main attributes and components of each AttributeRule that fall within the scope of this thesis are as follows (Chipman et al., 2016, pp. 19-20): -

- @AttributeName: describes the IFC schema's name for the attribute, connection, or inverse relationship.
- @RuleID: identifies the rule to be used as a reference at template rules created within concepts, where this rule is applied with a specific set of parameters. Each RuleID must be distinct from the others within the scope of its usage. However, the same RuleID can be utilized if they are used in different scopes; for example, two AttributeRules are defined within separate EntityRules, one for IfcPropertySingleValue and the other for IfcPropertyEnumeratedValue.
- EntityRules: a collection of EntityRule, which identifies the type of the AttributeRule. Any type may be used if the list is empty, as indicated by the schema.

When there is more than one specified entity, instances must correspond to one of them.

- Constraints: a group of expressions, each of which must result in the referred property being TRUE. This implies a Boolean AND combination. The Constraints element is of Constraint type. Constraint specifies a restriction on an attribute or entity that may require the value, type, or collection size to have equality (or another comparison) to a literal value or referenced value.

EntityRules has a set of EntityRule elements, which in turn may also have AttributeRules element, and so on, forming a tree structure. Each EntityRule has the similar elements as AttributeRule: @EntityName, @RuleID, AttributeRules, and Constraints. Added to this, EntityRule also has an optional element, References, which can refer to a partial ConceptTemplate. To guarantee that RuleIDs of partial templates are distinct within the context of their usage, the attribute "IdPrefix" is optional. All referenced RuleIDs begin with this attribute as a prefix. (Chipman et al., 2016, p. 20)

Therefore, there are two ways to refer to another entity and its attribute in a concept template. The first one is to refer to partial concept templates in the main concept template using the UUID attribute using @ref/@href. The second method uses the tree structure of attribute and entity rules.

The second main element in mvdXML is ModelView, which represents the description of an MVD. It contains zero-to-many ConceptRoot elements and references zero-to-many applicable ExchangeRequirement elements. The element ModelView has (Chipman et al., 2016, p. 21): -

- @applicableSchema
- ExchangeRequirements: list of the exchange requirements defined in this model view. They ought to show up in logical sequence. The ExchangeRequirement element describes the Exchange Requirement Model (ERM), which identifies the ER specified for a specific exchange scenario that are covered by the MVD. @applicability attribute determines if the ERM is applicable to import, export, or both.
- Roots: list of all root concepts that are defined within the scope of the model view.

Each ConceptRoot has the following attributes and elements (Chipman et al., 2016, p. 22): -

- @applicableRootEntity: identifies the IFC entity for which the concepts apply. It describes the class or data type of the instance being described or validated.
- Applicability: Applicability: a collection of TemplateRules based on a concept template and explains the circumstances in which the concepts apply to the applicableRootEntity. Before checking the TemplateRules applied to the concepts, those criteria must be valid. Applicability has only two elements: Template, which has a @ref/@href attribute that refer to a concept template by its UUID, and TemplateRules, which defines the rules by @Parameter attribute that refer to the RuleID of the reference concept template.
- Concepts: a set of concepts.

A Concept element defines the usage of a given entity with regard to the rules that must be followed. Again, each concept has Template and TemplateRules elements. It also has a Requirements element, which consists of a list of requirements outlining how the concept can be applied to specific import, export, or both exchanges (Chipman et al., 2016, p. 23).

TemplateRules Element can define a tree of logical expressions. The @operator property logically interprets each individual TemplateRule, which are grouped together under the TemplateRules element. The rule grammar used to define any rule in TemplateRule element is composed of: a metric and an operator (Chipman et al., 2016, p. 30). Table 2.1 and 2.2 describe the supported metrics and operators in mvdXML 1.1.

The most prevalent rule type is data existence, which is commonly a prerequisite for the other rules. It determines whether a schema-level "OPTIONAL" characteristic or relationship exists. Existence and cardinality rules can be defined in the Concept by the token "[Exists]" and "[Size]," respectively, in the rules formatted by the rule grammar. The second type ensures that specific elements/objects adhere to certain naming rules, particular attributes, and values. The "[Value]" token in mvdXML defines the data content rule type. It also has a number of operators, such as "Equal" and "Greater Than," as well as regular expressions for checking the value. The data uniqueness rule establishes an implicit constraint requiring unique attribute instance values. The "[Unique]" token in mvdXML defines uniqueness rules. The fourth rule is used to check the type of value that has been assigned to the attribute. The purpose of the conditional

rule is to ensure that the checking results of different rules are consistent. The rule is based on the findings of the previous rule types. Conditional dependence connections are currently represented in the mvdXML standard using the logic connectors "AND," "OR," and "XOR." (Zhang et al., 2015, pp. 30-33)

Table 2.1: Metric values description (Chipman et al., 2016)

Metric	Description
Exists	Describes the necessity of the existence of the attribute
Value	Describes the attribute's value
Size	Describes the size of a collection or string
Type	Describes the type of value assigned to the attribute
Unique	Describe if the value is unique within the Concept Template population of instances

Table 2.2: Operators description (Chipman et al., 2016)

Operator	XML Escaped	Description
=	=	Equal
!=	!=	Not Equal
>	>	Greater Than
>=	>=	Greater Than or Equal
<	<	Less Than
<=	<=	Less Than or Equal

2.3.4 MvdXML Versions

The first public release of mvdXML, version 1.0, was released in 2013. Providing a neutral structure for the documentation of MVDs was the primary goal of version 1.0. It was highly used to document the MVD ideas and concepts and describe the included attributes and entities to generate the IFC subset schema. The current development release, mvdXML 1.1, has replaced mvdXML version 1.0. The newer update focuses on validating and checking IFC file instances according to validation rules that were added to this version. The validation rules are based on the specified Exchange Requirements. In addition, the documentation process has been upgraded by including several new examples and fixing minor issues and inconsistencies found in mvdXML 1.0. This version will provide the foundation for all future additions and maintenance. MvdXML 1.2 is currently under development and is not yet considered a formal

buildingSMART standard. (buildingSMART, www.technical.buildingsmart.org/standards/ifc/mvd/mvdxml/)

2.4 Data Validation

2.4.1 Validation vs. Verification

According to EN 17412-1, verification is defined as "confirmation, through the provision of objective evidence, that specified requirements have been fulfilled, "while validation is defined as "confirmation, through the provision of objective evidence, that the requirements for a **specific intended use or application** have been fulfilled." Thus, validation varies from general verification because it has a specified intended purpose. In the thesis scope, validation is the core of this study since the goal is to validate an mvdXML instance file against a specified set of rules. BIM model validation involves determining whether a model has the necessary geometry and information. Its main objective is to make sure the models fit all standards for the use for which they are intended.

Thus, in other words, validation is a subjective procedure that depends on the authors and the use case. The concept of quality contains several criteria that might be evaluated subjectively, including "free from flaws," "suited for its purpose," and even "pleasant to the eye" or "customer delight and satisfactory/satisfaction." (Cunningham, 2013, p. 3)

2.4.2 Checking Types

A complete understanding of different types of checks is essential in this stage to clearly identify the validation process within this work scope. Research done by Hjelseth (2016) has been reviewed to determine the different types of checking and their use, which helped us to decide on the validation checking type as the core of this work.

Hjelseth (2016) divides checking concepts into the following types (Hjelseth, 2016):-

1) Compliance checking
    a. Validation checking
    b. Content checking
2) Design solution checking
    a. Smart objects checking
    b. Design option checking

Validation checking is the first type of compliance checking, which is the most common. Validation checking's principle is to ensure that the created solution adheres to the provided rules. To comply, the model's constraints must be within the rule's limitations. Codes, standards, contracts, best practices, and other established criteria can all be used to create rule sets. Clash detection, checking of component pairs, code checking, and checking based on information, are the most common types of validation checking. The goal of code checking is to see if the BIM solutions comply with codes, rules, and standards, among other things. The logic of compliance checking is based on the processing of established criteria with the following outcomes: "Pass," "Fail," or "Not checked." The "Not checked" result is due to the rule not being activated because of missing information. The rule set can include restrictions that specify value limits or the presence or non-existence of an item, and feedback will be shown with the result. (Hjelseth, 2016, pp. 356-358)

Model content checking is the second type of compliance checking, aiming to create an automated mechanism to ensure that the BIM's professional information is correct for a particular application. Content-based checking rule sets may help Information Delivery Manuals (IDM) and BIM guidance. Automatic content control in the BIM is a continuous activity throughout the design process. This type of checking is to compare the content and properties in BIM to a predefined list of relevant data. The checking has two alternatives: identified or not identified. Further actions can be writing a report to document compliance, and specific data can be erased or replaced with default values. This verification method ensures that the model has the exact amount of required information in the BIM, i.e., not too little or too much information. This is done by checking whether the model conforms with a requirement list of data specified based on IDM or Level of Information (LOI). (Hjelseth, 2016, pp. 358-360)

Design solution checking is usually done as part of the design process as a continual assistance. It is divided into smart object checking and design option checking. The former lets the object observe its surroundings and adapt to them using pre-programmed rules or algorithms. The latter enables the designer to evaluate a broader range of possible realistic options based on rules that recognize a predetermined circumstance and then conduct a search, based on a knowledge database, for appropriate options. The database might be an internal database or an archive of design solutions, or it could just be a set of search rules for relevant solutions. (Hjelseth, 2016, pp. 360-362)

Table 2.3 summarizes the classification of model-checking concepts according to Hjelseth (2016).

Table 2.3: Classification of checking types (Hjelseth, 2016)

Concept group	Concept type	Purpose	Outcome	Examples
Compliance checking	Validation checking	Validation	Pass/fail	Clash detections Code compliance
	Model content checking	Content	A filtered list	Exchange requirements
Design solution checking	Smart object checking	Adaptation	A modified model	Size of building parts related to the building
	Design object checking	Guidance	Options and advice	Knowledge database for relevant solutions selection

This section described different types of concepts of data model checking. It has been identified that the checking type related to this paper's scope is validation checking. Validation checking is the purpose of this paper, to check mvdXML against a specified set of rules. If the mvdXML adheres to the rules sets, then it passes all the checks, and it can be said that the file is valid.

2.4.3 XML Verification and Validation

Since mvdXML is also an XML file, this section discusses different methodologies employed in the industry regarding verifying and validating XML files. Several tools can generally check XML instance files against the schema, such as XML Schema (such as XSD) and Document Type Definition (DTD). Those XML schema languages can be used as verification tools to ensure that an XML document has correct syntax and structure as well as legal elements and attributes. Nevertheless, since schema verification is out of this thesis's scope and the focus is going beyond schema verification, Schematron as a validation tool is also reviewed in this section since it has more capabilities than any other XML schema parser.

2.4.3.1 XML Schema Definition (XSD)

The World Wide Web Consortium (W3C) has recommended using XML Schema Definition, or XSD, to describe and verify the structure and content of an XML document. It is mostly used to specify what kinds of elements, properties, and data the document can contain. To ensure that each element, attribute, and data type in the document adheres to its description, the information in the XSD is used as a check. Similar to older XML schema languages like Document Type Definition (DTD), an XSD offers more control over the XML structure and is, therefore, a more potent substitute. (W3C, www.w3.org)

2.4.3.2 Schematron

In simple terms, Schematron is a very versatile rule-based schema language that can express restrictions differently than XML Schema and DTD. Hence, using XPath and XSLT (Extensible Stylesheet Language Transformations) expressions, Schematron can be a very effective tool for validating XML data models due to its flexibility. (Tennison, 2004, p. 628).

Elements and attributes make up XML documents. Schema verification was the only checking method initially used for XML documents. This signifies that an XML document was considered legitimate if it had successfully undergone schema verification. Schema verification guarantees document structure but cannot check conditional and integrity requirements. Unlike grammar-based schemas like XSD, Schematron can attain the required level of validity. It can check both the structure and content of XML documents, in which constraints on entities can be defined, as well as demand on the presence of attributes can be applied. (W3C, www.w3.org)

Schematron can be considered as a potential alternative to mvdXML for checking mvdXML. However, it is somewhat ambiguous to choose it and presume it is more appropriate than mvdXML because no research has been done on Schematron validating IFC or any other equivalent, notably mvdXML.

2.4.4 Why Data Validation

Generally, the concept of testing and checking is widely established in the BIM industry to ensure that a deliverable unit, such as a BIM model, is suitable for specific use. For example, to ensure that a building model is suitable and correct for quantity takeoff and cost estimation. Therefore, BIM instance models are special models whose correctness and accuracy are vital because they are used in further operational steps. BIM model validation is thus a crucial procedure that ensures that the models have the required quality and are compliant with their intended purpose, in which a set of procedures is executed. The entire design process should involve model validation checks. By doing so, the models' quality is continuously monitored, and the necessary modifications can be made at the right time, avoiding errors during development. Continuous model validation can remove errors from the process as soon as feasible, minimizing the potential effects of future corrections.

Additionally, model checkers are frequently employed to validate critical systems or data models, which are then used in further other processes. The challenge lies in the fact that any model checker, like any other piece of complicated software, is prone to errors. As a result, given the importance of model checkers, explicitly verifying them, checking their accuracy, and enhancing their quality are critical. The model checker’s correctness must be ensured to trust the verification result. Although model checkers are exceedingly difficult to verify because their logic is always complex and highly optimized, the primary strategy of formalizing the development and validation of a formal verification system to increase user confidence in them is vital. (Sun et al., 2010)

Therefore, without implementing any checking techniques on model checkers, the model checker implementation must be assumed to be correct, which may not be true. Since the encoded rulesets used in the checking procedure are one of the main components of any checker, the correctness of those rulesets is one of the most critical procedures to guarantee the checker's correctness itself. Within this thesis scope, mvdXML is the data model with the encoded rulesets and must be checked.

2.4.5 How to perform Data Validation

One of the critical issues in this research is determining which types of checks are required and which rules should be set as well as how to create and implement them to check mvdXML. The types of required checks should be identified for an efficient validation procedure, and the general workflow steps should be followed. This section

discusses how data validation can be performed and how to tailor it to mvdXML, including validation steps, checking types, validation parameters, and dimensions.

2.4.5.1 Data Validation Checker and Encoded Data Formats

Since this work focuses on mvdXML, the IFC validation workflow using mvdXML is reviewed from several related works, which are discussed in depth in Chapter 3. Five main steps are generally implemented to ensure that received data comply with established requirements, shown in Figure 2.6. First, the appointed party defines data and information which are required to be present during the handover as Exchange Information Requirements (EIR). The requirements are then interpreted to model the data in a format that has been agreed on, as well as to encode the checking rules using a specified exchange format. The rules are then used to check the provided data from the BIM model automatically, and a report is generated with the identified findings.(Jaud & Muhič, 2022; Jiang et al., 2019)

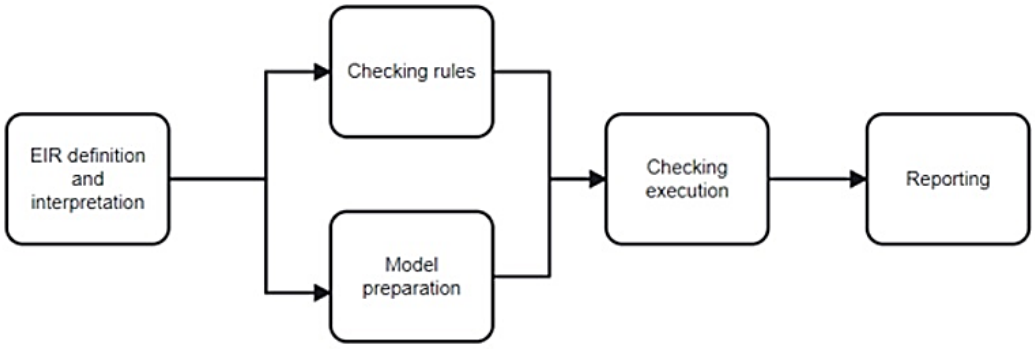

Figure 2.6: General validation procedure of IFC using mvdXML (Jiang et al., 2019)

A data model checker is a tool that checks the correctness of a system (Mosleh et al., 2016, p. 2). As shown in Figure 2.5, any checker tool has two inputs: the input file to be checked and the rulesets encoded in a chosen encoded data format. The first and second input files in this work to be checked and to check are mvdXML. Aside from mvdXML being selected as the format that encodes the checking rules, other similar available data formats are discussed and compared in this section to identify the reasons behind choosing mvdXML specifically to check mvdXML.

There are many possible approaches to encode rules based on exchange requirements, making it challenging for users to select the best one. Tomczak et al. (2022) classify and compare several exchange formats. Some of them, including documents

and spreadsheets, Product Data Templates (PDT), Data Dictionaries (DD), Information Delivery Specification (IDS), and mvdXML, are reviewed in this section.

One of the most used Information Requirements (IR) techniques today is text-based documents (DOC) due to their versatility, adaptability, and ease of use. Typically, they are created using popular text editors, enhanced with photos and diagrams, then assembled into Portable Document Format (PDF) files. Using spreadsheets (XLS) to define IR is another more organized method. It forces a tabular arrangement and makes computer parsing simpler. (Tomczak et al., 2022, p. 4)

Moreover, the IDS is a standard proposed by buildingSMART for describing BIM information requirements in a machine-readable and human-readable format. The primary purpose of IDS is to provide a straightforward yet complete method for authoring and validating information requirements. It is a crucial component that can be utilized as a contract to ensure that the correct information is delivered. Furthermore, it can build customized, use-case-specific requirements dependent on the project. One can specify what data must be included in the BIM model and verify that it is supplied with IDS. Each IDS definition has two parts: applicability (which filters the elements) and requirements (which specify what information should be given). Another popular format that is the core of this thesis and has been discussed thoroughly in Chapter 2.3 is mvdXML. The original intent of MVD was to narrow the considerably wider IFC to smaller subsets to serve a particular use case. Later it has also been used in validation procedures for IFC files. (Tomczak et al., 2022, p. 5)

In general, mvdXML and IDS have common features and similarities since they are in the same category. IDS, like mvdXML, make use of the XML markup language. Regarding the validation process, the two formats support IFC files and bring validation of IFC to the client. Nevertheless, only openBIM conforming tools can use IDS because it only applies to IFC entities and associated properties, unlike mvdXML, which is a generic framework that can be used for other data models than IFC. In other words, IDS is strictly dependent on the IFC schema. Moreover, IDS has a limited list of what can be requested. (Tomczak, 2022, pp. 5-10)

There are also several more possibilities for encoding exchange requirements, such as PDT and DD. Product Data Templates, also known as Data Templates, is a common standardized tabular set of product properties. PDT was developed to make it possible to compare similar products since it is possible for the involved parties to

communicate efficiently by using the same templates. PDT's emphasis is only on "what," not "by whom," "for whom," "when," and "for what." In response to ISO 23386's requirement for meta-information attributes, Data Dictionaries are established to define a standard structure to store information and objects along with their properties and relationships. This contains citations to outside sources as well as details on who made the things and when they were made. (Tomczak et al., 2022, p. 5)

The findings reveal that no single solution can address all the issues. Despite overlaps, each approach has a distinct function, advantages, and disadvantages and should be chosen carefully. All the strategies presented have a purpose and serve a vital role in the ecosystem. Within this paper's scope, mvdXML is the data format selected for encoding the checking rules for multiple reasons. Aside from the fact that one of the aims of this paper is also to analyze the capabilities of mvdXML in a validation process, it is also a naturally wise decision to pick it as a solution for validating mvdXML instance files. This is because it is nowadays one of the best and most used solutions to encode MVD since it almost captures most of its information (Chipman et al., 2016). Additionally, in terms of expressiveness, from the above-listed possibilities, the only format that can compete with mvdXML is IDS. MvdXML and IDS are more capable of a semi-automated validation procedure than the other listed possibilities (Tomczak et al., 2022, pp. 8-9). In other words, the coding effort to extract the stored data and requirements in an mvdXML or idsXML is less than that to be done in all other formats on the list to an automatic procedure. Therefore, considering the IDS option, a comparison of the similarities and differences between mvdXML and IDS has been made, and the results are outlined in Table 2.4.

Table 2.4: Comparison between mvdXML and IDS (Tomczak et al., 2022)

Functions		mvdXML	IDS
Functions	Validation	✓	✓
	Documentation	✓	✓
	Clash detection	x	x
	Precise filtering	✓	✓
	Geometry representation	x	x
	References	x	✓
Rulesets	Based on exchange re-quirements	✓	✓
	Develop new rulesets	✓	✓
	Combining rules	✓	✓
	Modify existing rules	✓	✓
Import	IFC files	✓	✓
	DWG	x	x
Reporting	BCF	✓	✓
	Visualization	✓	✓
Language	XML	✓	✓
Organization	BuildingSMART	✓	✓
Usage Scope	Schema-depandent	x	✓
	Generic	✓	x

There are many similarities between the two methods, mvdXML and idsXML. However, mvdXML is chosen over IDS for one important reason, as shown in the last feature of the comparison; IDS is limited to IFC entities only. Therefore, it cannot be used for any other data model than IFC; thus, it is completely excluded from this work as a solution. Besides, although IDS can be used for IFC validation, the most common method used while checking IFC files is mvdXML, which proves the relevance of mvdXML in validation procedures. It is of this thesis interest to analyze and examine mvdXML capabilities to be more versatile.

2.4.5.2 Validation Dimensions

Data quality is the extent to which a set of data characteristics satisfies requirements in ISO 9000-3:2015. The key issue to be addressed in this subsection is how to determine the quality of a given set of data and to which extent it overlaps with the mvdXML scope. The findings helped in identifying which exact validation checks should be set and which rules defining those checks should be identified based on the purpose.

Data validation is referred to as the study and use of various assurance techniques, standards, criteria, and systems to ensure data quality in terms of a set of quality parameters. Several general validation parameters can be used to check the input data. A Data Quality Dimension is a feature or component of data that provides a method for assessing and controlling data and information quality (Sidi et al., 2012, p. 302). Table 2.5 presents some key data quality parameters and their definitions from the literature. The green-colored dimensions are the ones that gain relevance in the upcoming paragraphs and are within the mvdXML scope.

The first parameter is data accuracy and correctness. This indicator refers to correct records and values that can be relied upon as a reliable source of information, measured by determining how closely a data value resembles another value, designated as accurate. Accuracy and correctness rules can specify that the values must be within a valid range or equal/not equal to a specific value. The second indicator is data timeliness, which determines how current data is. It is an important data quality attribute since out-of-data information can lead to individuals making poor decisions. Timeliness rules can set limits on a data value's lifetime, signaling that it should possibly be updated. Data completeness is another vital trait for measuring the extent to which the information is comprehensive. In other words, it ensures that all the required data is present and detects any missing values. In addition, data consistency is a concept that may be used to assess the consistency of data sets from several perspectives, such as checking that the information is of the correct type and format. It also ensures that data is entered in a logically consistent manner. Accessibility metrics can be used to assess how easily certain data can be accessible. Furthermore, a uniqueness check may be required for a specific field or value to ensure that an undesired duplication does not exist. (Gao et al., 2016, p. 434; Sidi et al., 2012, pp. 302-303)

Table 2.5: Summary of validation dimensions. The green-colored parameters are the relevant dimensions within this work's scope (Sidi et al., 2012)

Validation Parameter	Definition/Function
Accuracy and correctness	Checks how close a data value is to another regarded correct value
Timeliness and currency	Checks the degree to which the data's age is adequate for the task
Completeness	The degree to which the data available are enough in terms of breadth, depth, and scope for the task
Consistency	Checks that data is given in the same specified type and format and is compatible with preceding data
Accessibility	Ensures that the information is readily available or retrievable
Uniqueness	Ensures that the data is unique and not undesirably duplicated
Security and privacy	Assess the security and privacy of data sets from several angles
Relevancy	To what extent is the information relevant and valuable for the task

Without delving too far into the general validation rule types, mvdXML has specific checking capabilities. The following are the different types of validation rules in mvdXML (Zhang et al., 2015, pp. 30-31):

1. Data existence and cardinality, including the presence of attribute values and referred entities, as well as the size of collection data types
2. Data content, such as the value of simple data types and collection types
3. Data uniqueness
4. Data type
5. Conditional dependency, examining the conditional if-then dependency and consistency between them

More description of the mvdXML rule grammar can be found in Section 2.6.3.1.

3 Related Work

There has been sufficient research on several applications using mvdXML and how mvdXML could be applied in many ways to facilitate the automatic validation of different data models, the most common IFC files. Nevertheless, no researchers have considered the case of mvdXML checking itself or any other data model than IFC. This chapter describes the use cases of mvdXML in the current research studies, specifically mvdXML as a validation tool, and analyzes different checking guidelines and systems. The chapter concludes with a case study of the checking system validating itself.

3.1 MvdXML Use Cases

Several researchers have been working on mvdXML-related projects. Some examples that support the use of mvdXML are described in the following sections to demonstrate its various use cases. First, a study performed in the field of BIM filtering is reviewed. Second, mvdXML is introduced and examined for validation purposes, presenting two different studies.

3.1.1 Filtering

Baumgärtel et al. (2016) introduced a unique method in which mvdXML can be utilized to not only check BIM data, but also filter them. While model views are mainly required and used for documentation and validation, the need for sharing reduced building information models between participants is becoming more apparent (Baumgärtel et al., 2016). Zhang et al. provided the validation process, discussed in depth in Section 3.1.2 (Zhang et al., 2014). Moreover, other methods and checkers based on mvdXML have also been developed. Filtering, however, is not currently available with any of these tools. As a result, the mvdXML converter and ifcQL processor tools have been created in this method to enable both MVD-based validations and filtering (Baumgärtel et al., 2016). The main focus in this section is on mvdXML being used as a filtering tool.

The IFC handles several aspects of the construction industry. While this is beneficial for interoperability, it can cause issues with the level of detail in each domain. Therefore, nowadays, MVD-based filtering is becoming more critical. A two-step conversion for this study's implementation was applied. It begins with converting the contents of an mvdXML file to IFC Query Language (ifcQL) commands by a converter. IfcQL can

create, read, update, and delete BIM data easily. After that, the user must choose whether the result should be utilized for validation or filtering. In the second stage, the tool ifcQL processor is employed. The query commands extract the required information from the original IFC and create a simplified and reduced building information model. The use of various types of filters or filter layers is needed to achieve the necessary data reduction with model views. Four filter types were presented by Windisch et al. (2012), namely, schema-level filtering, class-level, object-level, and reasoning-level filtering. The simplified building information model can then be exchanged with one or more software applications. (Baumgärtel et al., 2016; Windisch et al., 2012)

3.1.2 Validation

MvdXML has always been highly used in the validation process in different scenarios. Five steps are generally performed to check the rules in mvdXML on IFC instance files (Jiang et al., 2019, p. 8). Figure 2.6 in Section 2.4.5.1 depicts those five main steps: -

1. Defining Exchange Requirements and interpreting it
2. Generating IFC models using BIM software based on the EIR
3. Converting the stored EIR to mvdXML format using an mvdXML rule transformer and inserting additional information using IfcDoc
4. Checking IFC models against mvdXML files
5. Generating BCF report

Several types of research have been done in this area of developing different types of checkers to validate IFC data models using mvdXMLs. In this subsection, two of those studies are briefly inspected. Both studies have been implemented on IFC models following the same general validation procedure mentioned above.

In a study carried out by Jiang et al. (2019), a strategy to manage green building information was created by merging mvdXML with semantic technologies (Resource Description Framework (RDF) and Web Ontology Language (OWL)) for green construction code validation. Since there are limitations of existing code-checking methodologies, this proposed method provides a solution for green construction code checking. The authors employed two BIM operating standards for green construction code validation in this work, mvdXML and Building Collaboration Format (BCF). Following the main general steps, a checker was created using the open standard mvdXML to define code validation rulesets and check IFC models against it. A BCF was then used to

report the checking process using a BCF generator, which can be shown in a BIM software tool, such as Revit. The rule sets are defined in Excel in a structured way and then imported into an mvdXML rule transformer to generate them in the mvdXML file. Once the mvdXML ruleset is completed, the IFC model can be validated. The mvdXML checker captures each created issue in a BCF report when the checking is completed. (Jiang et al., 2019)

Another important study is the implementation of a model view checker prototype by Zhang et al. (2014). This checker was built using the open mvdXML standard for validating instances of IFC models so that receiving systems derive the appropriate information. The mvdXML standard version 1.1 is used in this implementation, and the procedure has taken almost the same flow as the one mentioned above. The first step of this procedure is defining validation rule sets by interpreting exchange criteria and organizing validation rulesets, then executing the check and generating a report of the results. Here are again two open standards utilized in this implementation: mvdXML for structuring the validation rules and BCF for generating reports. IFC instances and mvdXML files are the checker's input, while BCF files are the output. The rules should be evaluated as true; otherwise, if an object instance breaks one of the rules, the application will raise an issue captured in the form of a BCF report. The mvdXML validation rules are structured based on two BIM operational standards, the Dutch Rgd BIM Norm and the Norwegian Statsbygg BIM Manual. The prototypical checker is then tested with example use cases. This study has identified that the mvdXML standard and its implementation may be used as a light checking tool for IFC validation even if there are still some general issues and restrictions. (Zhang et al., 2014)

3.2 Self-Validation

The next generation of software nowadays is self-validating. Self-checking system design has gained interest in the past, intending to create tools capable of independently detecting the occurrence of a flaw during routine operational life. Developing innovative self-validation techniques and quality-control systems is highly important for more efficient construction processes. (Bolchini & Salice, 2000; Sun et al., 2010)

Unfortunately, no work has been found in the field of BIM and the industry of AEC related to self-validation methods. To our knowledge, till these days, there are not any other similar methods. However, a study carried out by Sun et al. (2010) in which an approach for checking model checkers is proposed and applied. The authors present

a method that combines code contracts and model-checking procedures to systematically verify Process Analysis Toolkit (PAT) and increase its quality, a real-world model checker. PAT is a self-contained framework for designing, simulating, and verifying real-time systems. It uses a variety of model-checking techniques to verify different properties. (Sun et al., 2010, p. 519)

PAT is designed to be an expandable and modular framework allowing users to create customized model checkers quickly. It offers a library of model-checking algorithms to verify a wide range of systems. A suitable verification algorithm is invoked depending on the property to verify. Since the library of the algorithms is shared by all the created modules, the accuracy of PAT is thus critical and must work correctly and efficiently. Therefore, it has constantly been thoroughly tested, but issues and bugs are still reported now and then. The leading cause of these bugs is that PAT is continually being updated because coding modification and extension are frequently made to achieve efficiency, resulting in additional discovered bugs. PAT now has over 800 registered users from over 180 different organizations. In conclusion, PAT has grown into an extensive software package requiring systematic quality control. (Sun et al., 2010, pp. 521-522)

This approach is unique because it focuses on model checking the model checker itself. In short, a checker that checks itself, i.e., PAT is checked using PAT. It consists of two main components: code contracts and model checking. The contracts serve as a validity specification, and model checking is a valuable tool for looking for contract violations. The validation process involves specifying the code contracts relating to the complete system as a property and then formally verifying via model checking. The models can be saved as PAT models, allowing us to model verify them directly in PAT itself. Integrating code contracts with the model checker is considerably broader and can be used to check a wide range of C# software systems. (Sun et al., 2010, pp. 520-531)

Full details and specs of the method can be found in (Sun et al., 2010), which are not more discussed in this paper because it does not overlap with the scope of BIM and mvdXML. Despite this, this study case was sufficient to be the proof that gave us the insight that our approach is applicable and necessary for real-world problems since it can be used in a broader scope also.

3.3 Summary

To conclude this chapter, the following main points are captured from the prior work: -

- The general checking procedure is outlined in five main stages for almost all the tools and techniques: -
  1. interpretation of exchange requirements
  2. generating data model
  3. structuring validation rulesets
  4. check execution and
  5. report generation
- Since mvdXML is used for validation purposes, it is critical to validate it to have confidence in its outcome
- The future is about self-validating, which has been proven to be applicable and more efficient for real-world challenges in several other fields

Besides, as clearly discussed, there is research gap in two crucial aspects. First, no prior self-validation methods were considered within the scope of BIM, IFC, or mvdXMLs. In addition, the case of using mvdXML to validate any other data model than IFC was not also properly considered, although mvdXML is a generic framework that can be used on other data models. As a result, existing approaches cannot be directly applied to checking any kind of data models. Thus, mvdXML use has always been looked to as a validation method for IFC only from the researchers' point of view. To ameliorate this situation, the development of new technology is necessary to better the current situation. However, since there is a huge gap in this area, we found out that we also cannot directly apply our approach. Thus, the priority of this thesis's goals has been restated. The main goal would be is to generally analyze the capabilities of mvdXML to validate other data models than IFCs to find out how close or far is it from being a versatile validation tool. Along with this main aim, the mvdXML itself will also be checked with a robust prototypical validation method that can be further developed.

Also, many factors are involved in object or data model compliance checking. Each language or tool has its own function and scope; therefore, it is critical to decide consciously which to use based on specific purposes. After reviewing the theoretical background of this paper's problem and the literature review, it has been found that mvdXML significantly needs a method to be checked since it is highly used in the validation procedure of other data models, such as IFC. However, no method has been

proposed until now in the literature review to check mvdXML itself, even though some discussed issues have been discovered in some mvdXML instance files such as ReferenceView mvdXML. Furthermore, mvdXML is used to check itself in this work since it was successfully used in validation procedures of IFC and can be used on other data models since it is already a generic framework. Moreover, mvdXML is the most suitable method for encoding MVD itself, as recommended by bSI. Hence, using mvdXML to check mvdXML is more closely the proper procedure. Other alternatives of data formats and their exclusion have been discussed in Section 2.4.5.1. Besides, Schematron, reviewed in Section 2.4.3.2, can be considered another potential method other than mvdXML for checking mvdXML, since it can already check XML files, in general. However, since no research has been carried out previously on Schematron validating IFC or any similar, particularly mvdXML, it is somehow vague to choose it and assume it is more suitable than mvdXML.

4 Methodology and Implementation

This research aims to test the practicality of checking a data model other than IFC files using mvdXMLs. It is essential in this paper to discover what is still required for an mvdXML to be a versatile tool for many different functions and activities.

In the scope of this thesis, the Design Science Research Methodology (DSRM) is employed, as established by Peffers et al. (2007). DSRM includes six main steps: -

1. Problem identification and motivation
2. Definition of the objectives for a solution
3. Design and development
4. Demonstration
5. Evaluation
6. Communication.

The following sections discuss in detail the methodology steps, as well as the datasets used for the evaluation and their sources.

4.1 Aims and Objectives

This study aims to analyze the capabilities of mvdXML critically. The analysis must be carried out according to a specific use case. The use case chosen for this work is to analyze mvdXML as a validation tool for a data model other than IFC. The data model selected to be checked is mvdXML since the secondary goal of this thesis is to ensure the quality of any generated mvdXML. Thus, mvdXML is used to validate itself. The following section describes the detailed approach of this study.

Alongside the main aim, this thesis has the following secondary goals: -

- To check the accuracy and correctness of mvdXMLs using mvdXML
- To enable validation of mvdXMLs that other end users create for specific use cases
- To allow end users to build and customize their checking mvdXMLs
- To enable mvdXML to be a versatile tool by checking a data model other than the IFC schema

- To discover the potential for expanding and enhancing the mvdXML schema in the subsequent versions

Data validation is a task that is typically carried out in many aspects and fields. This practice has been carried out for a long time, but despite this, methods and techniques have never been standardized (Di Zio et al., 2016, p. 3). This is because validation is a subjective process. Thus, the validation criteria rely heavily on the author's objectives.

Since validation criteria are subjective, this paper defines specific objectives for the validation process of mvdXML that are as objective as possible. Hence, the methodology can be later evaluated based on these objectives. Based on the aims mentioned above and expected outcomes, the following objectives are settled: -

- Detect missing optional elements that can be required in some use cases
- Ensure specific values for specific attributes
- Detect mistakes in attribute usage
- Detect alien elements in strings
- Detect contradictions
- Discover the capabilities and limitations of mvdXML

4.2 Approach

As previously mentioned, the general framework of this paper is the DSR methodology. The previous chapters and sections discussed the motivation, research question, solution, and objectives. This section demonstrates the implementation of the proposed solution. The working steps that have been done and are still to be done in this work can be summarized as follows: -

1. Review the state of the art of the mvdXML
2. Develop the self-validation procedure
    2.1. Define error scenarios that can be caused
    2.2. Define the checking rules
    2.3. Define the concept templates and concept roots of the checks
    2.4. Implement the procedure and test it
3. Analyze the mvdXML capabilities based on the self-validation procedure results
4. Propose solutions to expand the mvdXML schema capabilities in the following versions

The proposed approach tests the versatility and adept of mvdXML as a validation tool on a schema other than IFC by challenging it to check itself with several different checks to discover what is testable with this tool and what is not. Hence, in this paper, an mvdXML checker for mvdXMLs is presented. A prototypical implementation is demonstrated and tested using different example use cases.

Chapter 3 reviewed several related previous works in order to have a reference to an efficient and successful data validation procedure. It is necessary to create and follow a general framework for data validation. Previous related works inspire the validation process shown in Figure 4.1. It consists of four main steps; 1. interpreting the requirements and analyzing the errors that can be generated, which helps in 2. encoding the validation rule sets in an mvdXML, 3. executing those checks using the checker, and 4. generating a pass/fail report.

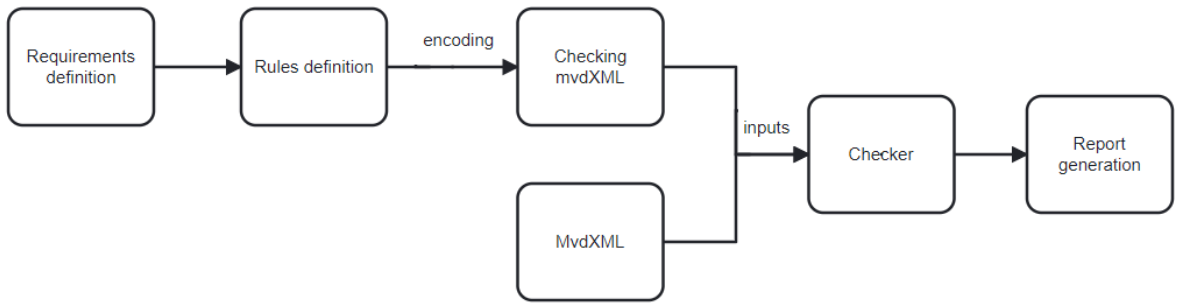

Figure 4.1: Methodology of the proposed mvdXML validation process

Since the core structure of the mvdXML is maintained in the proposed developed mvdXML, the logic behind the checking rules is kept. The checker checks schema conformance first. If the data model is schema conformed, it executes the mvdXML checks. Otherwise, the checker aborts and does not execute the mvdXML checks. If the mvdXML check is conducted, the process continues with checking the applicability. If applicable, the output also contains information about which entities pass and which fail, as shown in Figure 4.2. A BCF report was generated in most previous literature review cases. However, the proposed checker uses the Markdown format for the report. Markdown is a simpler report format that just reports fail/pass results. The logic behind the checking rules is to show first which elements are applicable and not applicable and then which of the applicable elements pass or fail. Thus, BCF was out of this thesis's scope since it has many more features than just reporting pass/fail results and has many more IFC-specific attributes.

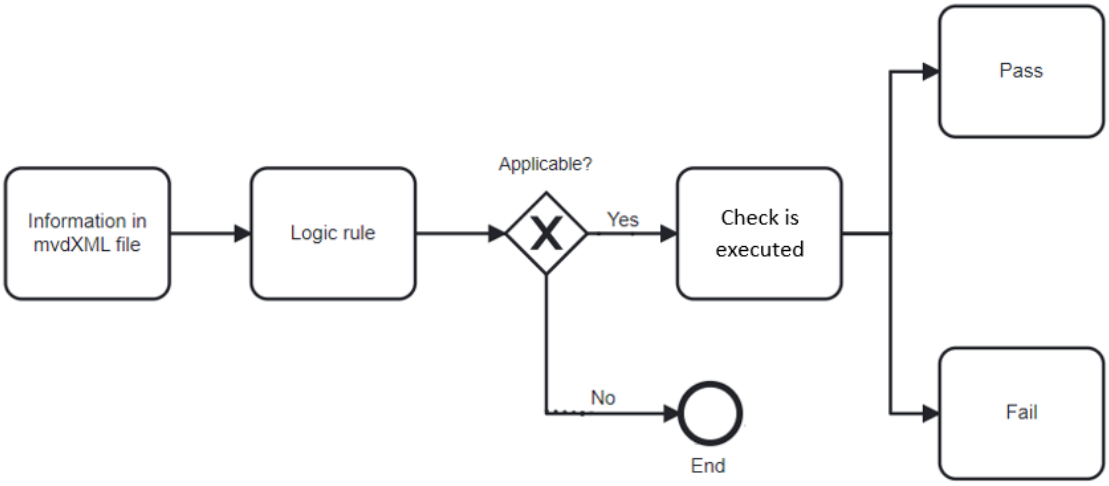

Figure 4.2: Illustration of the logic rule behind the mvdXML validation procedure

4.3 Evaluation Approach and Datasets

To attain the most accurate objective results and guarantee the results' correctness, the implemented method must be checked on several stages parallel to its implementation. So that the implementation can be continuously evaluated and always get early feedback after each testing step in the early development phase, which helps catch errors and improve the performance of the mvdXML checks faster.

Hence, before the main implementation of the mvdXML checks, the database for the continuous evaluation process must first be set up. The goal is to gather as many mvdXML instances as possible to test the method on different levels and to ensure precise, accurate results. Thus, the technique and the capabilities of mvdXML can be analyzed based on correct results.

The following sections cover in detail the four stages of the evaluation process, the source of the datasets, and provide a thorough explanation of how they were prepared.

4.3.1 Evaluation Phases

The evaluation procedure is divided into four significant stages. The first two are considered the main two tests in the very early design phase, which we depend on to give us quick feedback for adjusting the checker. The following two steps are then carried out after partially testing the mvdXML checks and are somehow expecting accurate results. These last two stages then come to implementation to maybe fix minor hidden issues and get final results, which will mainly be a large part of the analysis. Figure 4.3 illustrates the four stages and their relations.

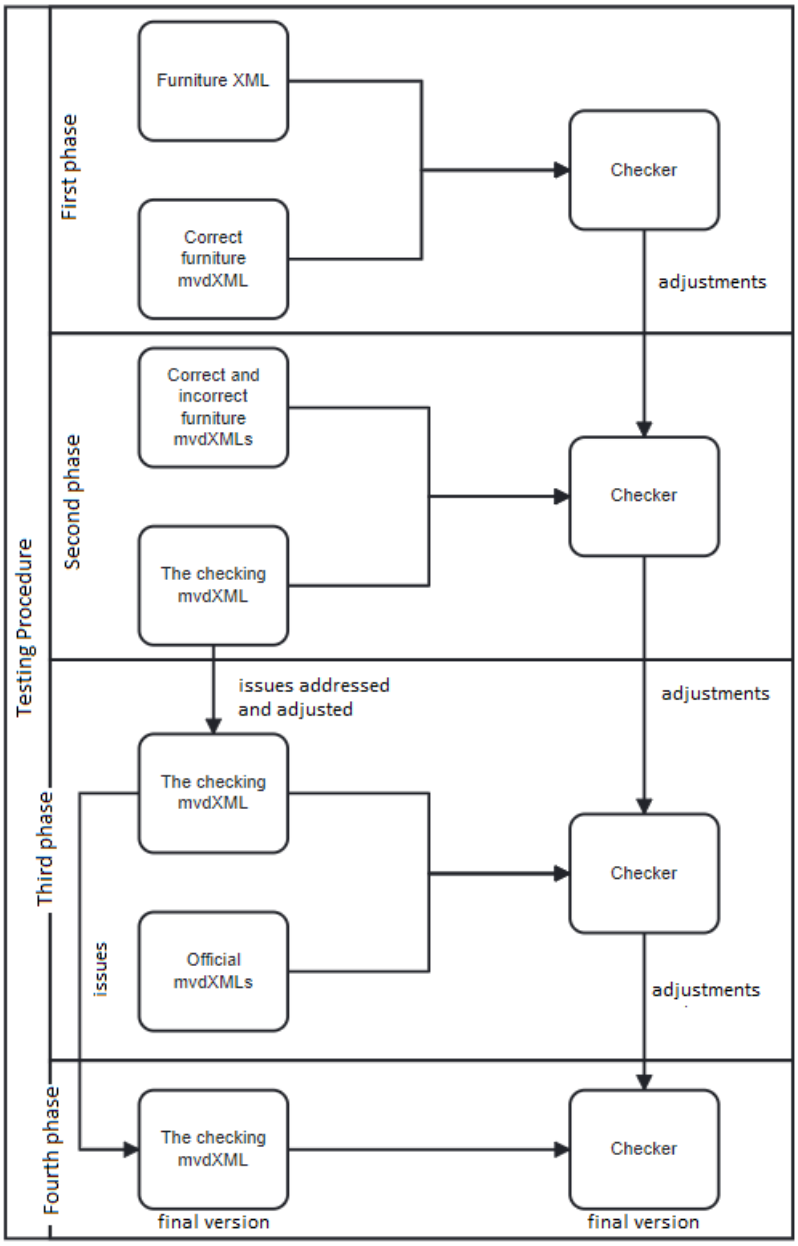

Figure 4.3: Four stages of the evaluation process of the proposed method

The first step is to check an XML file against an mvdXML file. Both of those files are created by the author and are explained in detail in the next section. Based on the feedback of this first loop, the checker is adjusted. This step helps better understand the checker process and its generated results and eliminates any errors. Next to this, the checker is used again also to check the author-made correct and incorrect mvdXML files against the primary checks encoded in an mvdXML file. The incorrect mvdXML files have intended errors to observe the ability of the method to capture them. This step helps improve the mvdXML checks along with the checker. Finally, the technique is used to check official mvdXMLs, and the final version of the mvdXML checks itself

also. These steps, especially the first two, keep running in loops, helping to optimize the procedure until satisfying results that we aimed for and expected are achieved.

4.3.2 Furniture Example

Gathering the dataset is a crucial step to guarantee the best correct performance of the mvdXML checker. Therefore, there are four different sources for the dataset in this study: a small example developed by the author, several official and other mvdXML files that are freely available on the internet, and the developed mvdXML checker itself. Those types of datasets are selected to be included in our evaluation.

The first dataset is a simple example developed from scratch by the author for several reasons. We are confident of the content of this example as we designed it ourselves. Hence, the expected results are already known, and the checker can then be evaluated initially to determine whether it works appropriately as expected. In addition, this example is a lot simpler than the mvdXML schema. It is easier first to identify implementation errors while testing it on a small sample instead of trying it automatically on more complex mvdXML files. Finally, we want to challenge the method with different scenarios and ideas that have different combinations and apply different combinations of checks on them to see how the checker responds. That is how we can test it and later analyze it based on how versatile it is, i.e., to test whether it can work on any possible scenario or not.

The designed example, the furniture example, consists of a collection of information related to a furniture class with different chairs. Figure 2.4 in Chapter 2 depicts the UML diagram of this furniture schema. It is composed of a furniture class, which has chairs. The chair class has three other classes: armrests, legs, and production. The three components, chair, leg, and armrest, inherit some features from a class called component, which has two objects of class material: cushion and structure. Based on this schema, a furniture dataset and mvdXML were created and used in the evaluation procedure.

4.3.3 Other mvdXMLs

Another source of the dataset for the third phase of the testing process is the official buildingSMART mvdXML instance files, such as IFC4 Precast, Reference View, and Quantity Takeoff, which are easily obtained from the official website (buildingSMART,

https:technical.buildingsmart.org/standards/ifc/mvd/mvd-database/). Other three official buildingSMART files have been obtained from other links to be tested: -

- Alignment Based ReferenceView: https://bsi-infraroom.github.io/IFC-Documentation/4_3_0_0/abrv/HTML/link/ifc4x3_rc4-update.htm
- IFC4 General Usage: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2/HTML/
- COBie: http://docs.buildingsmartalliance.org/MVD_COBIE/annex/annex-a/construction-operations/COBie.mvdxml.txt

Besides, the mvdXML example in the documentation of mvdXML 1.1 is employed (Chipman et al., 2016, p. 36). Moreover, this phase uses two models made by the Institute for Applied Computer Science (IAI) at the Karlsruhe Institute of Technology (KIT) to represent the case where mvdXMLs are created by other end-users for specific uses other than the official ones (Institute for Automation and Applied Informatics (IAI), 2021).

Figure 4.4 visualizes that seven of the files are official BuildingSMART mvdXML files, while the other two are from Karlsruhe Institute of Technology.

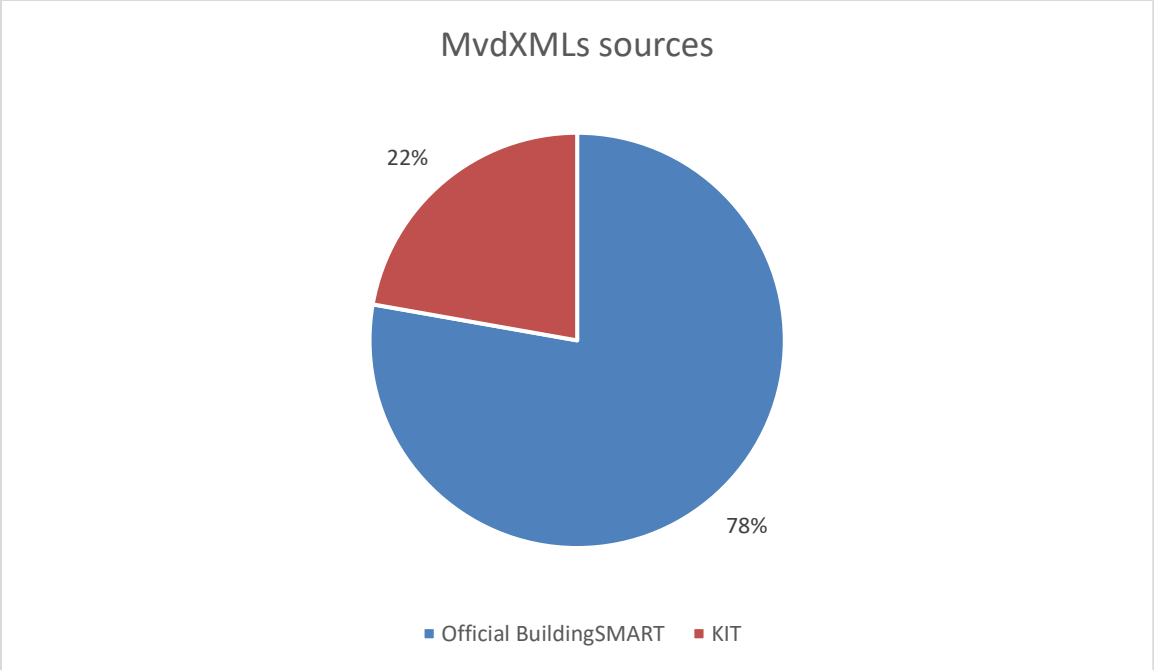

Figure 4.4: Pie chart mapping the number of the mvdXML files from each source

Finally, the last source is the developed mvdXML itself, i.e., the checking mvdXML is used to validate itself.

4.4 Design and Development

After gathering and preparing the datasets, the next step is to find out different error scenarios and design the checks respectively. This step analyzes the errors that can be generated in an mvdXML file based mainly on the documentation. The focus is primarily on the mistakes that can happen, but the schema checks cannot catch them. Furthermore, some of the check's ideas are derived from the recommendations, which are not enforced in the mvdXML schema. Other checks are subjective checks based on the author's opinion and not the error scenarios. The following sections cover the different types of errors and their rules.

4.4.1 Error Scenarios

All the errors that came up in this thesis, other than schema errors, are summarized in Table 4.1. We try to create as many scenarios as possible to challenge the mvdXML for several types of checks. These error cases are based on an objective basis and derived from the documentation of mvdXML 1.1 (Chipman et al., 2016), the general guidelines and checking procedures outlined in the theoretical background and literature review in Chapters 2 and 3, and the author's understanding of the topic. Objective basis within this context means without the inclusion of the author’s opinion. Based on those error scenarios, several types of objective mvdXML checking rules are designed, described in the next section.

There are six error concepts defined that can occur in any mvdXML file. The first error is RuleID not being unique within its scope of usage. When for example, two attributes have the same value for the RuleID attribute in one concept template, this check fails. This error may also occur if two RuleID(s) are similar in two different concept templates, while one of them is referred to as a partial concept template into another, and there is no unique IdPrefix. Otherwise, with the presence of a unique IdPrefix, in this case, the RuleID value can be duplicated. Therefore, errors 1 and 2 complement each other in the case of the reference to a partial concept template; either RuleID or IdPrefix must be unique to form a unique combination of the value. The schema restricts the type of RuleID and IdPrefic to mvd:ruleId, whose base is a normalized string. However, it does not have the feature of limiting the duplication of those attributes. RuleID, with or without the combination of IdPrefix, must be unique to prevent confusion when using it in the Parameters string. Figure 4.5 depicts a part of the “ComponentRuleIDUniqueness” mvdXML file (described in detail in Section 4.5.1), showing that the RuleID attribute is

duplicated twice for MaterialName and Category. Hence, this file fails against the RuleID check.

```
10      <Rules>
11        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
12          <EntityRules>
13            <EntityRule EntityName="Material">
14              <AttributeRules>
15                <AttributeRule AttributeName="Name" RuleID="MaterialName" />
16                <AttributeRule AttributeName="Category" RuleID="Category" />
17              </AttributeRules>
18            </EntityRule>
19          </EntityRules>
20        </AttributeRule>
21        <AttributeRule AttributeName="Structure" RuleID="Cushion">
22          <EntityRules>
23            <EntityRule EntityName="Material">
24              <AttributeRules>
25                <AttributeRule AttributeName="Name" RuleID="MaterialName" />
26                <AttributeRule AttributeName="Category" RuleID="Category" />
27              </AttributeRules>
28            </EntityRule>
29          </EntityRules>
30        </AttributeRule>
31      </Rules>
```

Figure 4.5: Section of “ComponentRuleIDUniqueness” mvdXML file shows an error of RuleID duplication within the same concept template

The third error is regarding the Parameters attribute. This error is generated if one of the following scenarios happens: -

a. No metric is defined within the Parameters strings: The user may forget to specify a metric or use the old grammar of mvdXML 1.0 even if the schema followed in the mvdXML file is mvdXML 1.1.
b. Presence of MetricValue other than the enabled ones: There may be a typo of [Exist] instead of [Exists], for instance.
c. Presence of any word that does not make sense: The only metric values that make sense for mvdXML 1.1 is defined in Chapter 2.4. Any other metric that might be thought of as supported or any other word between the square brackets instead of those five metrics generates an error.
d. Presence of an attribute that is not in the mvdXML file: Parameters string may have a RuleID that is not even defined in the referred concept template.

The schema generally enforces the existence of Parameters attribute for each TemplateRule. However, it did not interfere with the content of the string regarding those points.

Another four scenarios generate the “ref” error: -

a. No Template is referred to in the Applicability or/and Concept nodes
b. No Template is referred to in the EntityRule node when the References element exists
c. The @ref attribute in the EntityRule node refers to a wrong concept template UUID: This error can be caught by checking if the EntityName of the EntityRule is similar to the applicableEntity of the referred concept template or not. If not, an error is reported
d. The @ref attribute in any of the nodes mentioned above refers to non-existent ConceptTemplate

The first two errors and the fourth are already schema errors in which the proposed checker is tested on how it responds to such checks.

The fifth error occurs if two conditions exist. If there is a blank Parameters string in Applicability or Concept nodes, and no Constraint exists in the referred concept template. The last error scenario describes a case in which mvdXML, ConceptTemplate, ModelView, ConceptRoot, or Concept have an empty @Name attribute. The schema here enforces the existence of the @Name attribute since it is a required attribute, but it does not enforce the presence of a value for it.

Table 4.1: List of the possible error scenarios

Error Scenarios			
ID	**Attribute**	**Type**	**Error Description**
1	RuleID	Uniqueness	Two attributes have the same RuleID name within the same scope
2	IdPrefix	Uniqueness	Two IdPrefix have the same name within the same scope
3a	Parameters	Content Correctness (Value)	No MetricValue in the parameters string
3b			Presence of MetricValue other than the enabled ones in the mvdXML
3c			Presence of any word that does not make sense instead of a MetricValue
3d			Presence of an attribute that is not in the mvdXML file
4a	ref	Content Correctness (Value)	No template is referred to in applicability or concept node
4b			No template is referred to while the reference attribute exists in the EntityRule node
4c			The ref in EntityRule node is referring to a wrong concept template UUID
4d			The ref attribute in any of those nodes refers to non-existent ConceptTemplate
5	Constraint	Existence	The parameters string is empty, and there is no Constraint in the referred ConceptTemplate
6	Name	Content Correctness (Value)	Required attributes such as Name exist but are empty

4.4.2 Checks Definition

The proposed mvdXML validation approach shares similar features to the IFC validation approach. In addition to adhering to all of the syntactic restrictions outlined in the schema, a semantically validated IFC instance also needs to follow additional guidelines about the proper usage of model elements (Yang & Eastman, 2007).

Similarly, the scope of this thesis is not verification but validation. Based on the above error analysis and the author's understanding, several types of mvdXML checks are defined. The focus of the checks is not schema checks, but errors that can happen and are not enforced in the schema, i.e., schema checks are out of scope. Nevertheless, two schema-level checks and some subjective checks based on user-specific requirements are defined to review how the checker responds. The primary purpose of those checks is to challenge the mvdXML as much as possible to know its limits. Furthermore, to decide at the end whether and how to expand mvdXML's checking capabilities beyond the currently existing core checks of IFC data models, i.e., we are trying to be mean to mvdXML and push it to its limits.

As an illustration for IFC checks, consider Figure 4.6, which depicts a rule of "any IfcWall should be typed by an IfcWallType" in IFC 4. The usual way of defining IFC checks follows the same concept as this illustration. The concept template describes the root entity object (IfcObject) down to its relationship with the attribute value (which is here IfcTypeObject) through the tree structure of attributes and entities. The check itself is then employed in the concept by specifying the applicableRootEntity (which is Ifcwall in this situation, a subtype of IfcObject), and the check is defined on the attribute using any of the five metric values [Value],[Size],[Exists],[Type], and [Unique]. (Zhang et al., 2014, pp. 29-30). Some of the designed checks follow the same concept logic of IFC checks, and others differ slightly in the check definition. The mvdXML checker is tested using all those different types of checks, and the results are used to examine the versatility of mvdXML as a validation tool for other data models.

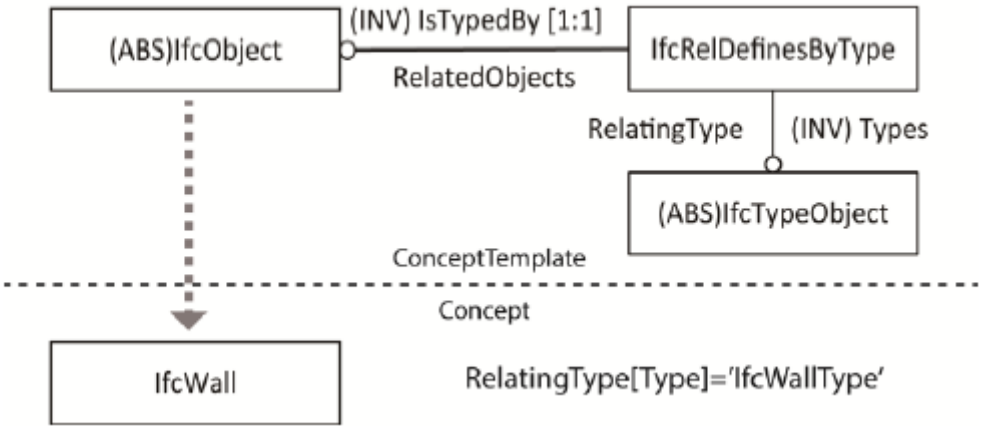

Figure 4.6: Model View Concept Example in mvdXML (Zhang et al., 2014)

Objective checks are based on the defined objectives in this thesis and on the facts drawn from the theoretical concepts, the literature review, and some of the discussed issues among the users (buildingSMART, https://forums.buildingsmart.org/t/problems-when-parsing-and-automating-referenceview-v1-2-mvdxml/1935). In other words, the objective checks are not influenced by personal opinions. They only consider and represent the current situation of mvdXML and their facts. In contrast, the subjective checks are influenced by our perception of a valid mvdXML file, i.e., personal opinions influence them. Despite this, some subjective checks may have an objective point of view in particular situations (discussed in the following chapters). However, these particular situations can be solved with other checks from another point of view.

Table 4.2: List of the objective checks. The results of the red-coloured checks are of interest

Objective Checks Definition				
ID	**Check Name**	**Check Description**	**Attribute**	**Metric Value**
1a	RuleID Uniqueness	RuleID is unique within the same scope	RuleID	Unique
2a	IdPrefix Uniqueness	IdPrefix is unique within the same scope	IdPrefix	Unique
3	MetricValue	There must be a correct applicable metric value for each attribute in every parameter string	Parameters	Value
4	ParametersAttributes	Each attribute existing in the Parameters string must have RuleID	Parameters	Value
5	TemplateReference	There must be a ref attribute in Applicability, Concept nodes, as well as EntityRule if References exists	ref	Exists
6	EntityReference	Check that the ref in EntityRule node is referring to the correct ConceptTemplate	ref	Value
7	ReferencedConceptTemplates	All referenced ConceptTemplates in the EntityRule nodes and ModelViews exist in the mvdXML file	ref	Value
8	Constraints	If parameters are empty, a constraint must exist in the referred ConceptTemplate	Parameters & Constraint	Value & exists
9	Name	Name attribute must exist in mvdXML, ConceptTemplate, ModelView, ConceptRoot, and Concept	Name	exists

Table 4.3: List of the subjective checks. The results of the red-coloured check are of interest

Subjective Checks Definition				
ID	**Check Name**	**Check Description**	**Attribute**	**Metric Value**
1b	RuleID Existence	Each AttributeRule has a RuleID	RuleID	Exists
2b	IdPrefix Existence	IdPrefix exists	IdPrefix	Exists
10	EntityRule	Each AttributeRule must have EntityRule	EntityRule	Exists
11	OperatorExistence	Operator exists when there is more than one TemplateRule	Operator & TemplateRule	Exists and Size

ReferenceConceptTemplates and TemplateReference checks are already schema checks and enforced in the schema, but it is crucial to examine how the checker will respond to schema compliance checks. Furthermore, as shown in the objective checks table, there are only three types of metric values used; Unique, Exists, and Value. We add another metric value in the subjective checks, “Size” in the OperatorExistence check.

It is essential to say that even the subjective checks may have an objectively valid reason behind them, according to the author. For example, the error "3d" in Table 4.1 above. The error, as discussed before, describes a case in which a parameters string contains a RuleID that does not exist, as shown in Figure 4.7, depicting this case in the “ComponentParametersAttributes” mvdXML file.

```
    <Rules>
      <AttributeRule AttributeName="Cushion" RuleID="Cushion">
        <EntityRules>
          <EntityRule EntityName="Material">
            <AttributeRules>
              <AttributeRule AttributeName="Name" RuleID="CushionMaterialName" />
              <AttributeRule AttributeName="Category" RuleID="CushionCategory" />
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="Structure" RuleID="Structure">
        <EntityRules>
          <EntityRule EntityName="Material">
            <AttributeRules>
              <AttributeRule AttributeName="Name" RuleID="StructureMaterialName" />
              <AttributeRule AttributeName="Category" RuleID="StructureCategory" />
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
  </ConceptTemplate>
</Templates>
<Views>
  <ModelView uuid="7feec79c-0c8f-45db-9db9-bcad6fcf6532" name="ComponentModelView" applicableSchema="FurnitureSchema">
    <ExchangeRequirements>...</ExchangeRequirements>
    <Roots>
      <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applicableRootEntity="Component">
        <Concepts>
          <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <Requirements>
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-7bf062a4dc46" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-7bf062a4dc46" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules operator="or">
              <TemplateRule Parameters="Category[Value]='Metal'" />
              <TemplateRule Parameters="StructureName[Exists]=TRUE"/>
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
```

Figure 4.7: ComponentParametersAttributes mvdXML file shows an error in the Parameters string

Since there are two sources of this error, it can have two ways to be checked. The first source, which is more general covering a broader aspect, is that the parameters might have an attribute that is not even defined in the mvdXML. In other words, the parameters string contains any word not defined as a RuleID in the mvdXML file and out of the scope, for example, “Category” in Figure 4.7 above. Respectively, the first objective way is to check that the parameters string only contains the attribute RuleIDs. So that any alien element within the string, such as “Category,” generates an error. This is the most accurate way to catch such an error. The second source, which has narrower scope but could be easier to implement within the mvdXML scope, is that the parameters might have a RuleID that was intended to be present in the mvdXML but forgotten, as shown in Figure 4.8, showing a part of “ComponentRuleIDExistence” mvdXML file. Therefore, the second subjective way (RuleID Existence) is to ensure that for each attribute, a RuleID exists. This method is another way to ensure that each referred attribute in the parameters string exists in the mvdXML file. Unlike the first check, this check catches that there is no RuleID, such as in Figure 4.8 but does not catch any error in Figure 4.7 because the RuleID itself exists. Thus, there is still a risk for any alien string to exist in the parameters string, which is not even an attribute

```
      <Rules>
        <AttributeRule AttributeName="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name"  />
                <AttributeRule AttributeName="Category"  />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name"/>
                <AttributeRule AttributeName="Category" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="979934da-8b06-4d07-ae9a-756e5918a66a" name="ComponentModelView" applicableSchema="FurnitureSchema">
      <ExchangeRequirements>...</ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applicableRootEntity="Component">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="Cushion[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>...</Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Structure[Exists]=TRUE" Description="check that structural material exists when cushioned material exists" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
```

Figure 4.8: ComponentRuleIDExistence mvdXML file shows an error of non-existence of RuleID

Moreover, some checks, such as RuleID and IdPrefix, complement each other. The base concept behind the two checks of RuleID uniqueness and IdPrefix existence is that we want to ensure that the RuleID attribute is unique within the same scope. Within the same scope means all concept templates which are related by referring to each other. Therefore, the RuleID for each attribute must be different unless there is an IdPrefix attribute on the EntityRule node of one of the AttributeRules that are similar in two different concept templates. To implement this within the scope of mvdXML, the objective RuleID uniqueness check is defined to ensure that each AttributeRule is unique within one ConceptTemplate. At the same time, the subjective check of IdPrefix existence guarantees that RuleIDs of referred concept templates are unique. Therefore, the RuleID being unique between different concept templates is assured because even if they have the same name, there is a prefix for the RuleID name of one of them.

Those are some examples of subjective checks based on an objective source. Other subjective checks are user-based and depend solely on the author's opinion, which may showcase some features. Such as the EntityRule check, which specifies that for each AttributeRule, there must be an EntityRule to know its type because, in complex schemas, there might be the exact name of two attributes but with different entities. Another subjective check is OperatorExistence. Although the schema specifies a

default "and" operator to the TemplateRules node, a modification in the schema of not defining a default operator along with this check must be considered. So that we prevent a case in which the user forgets to specify another operator than the "and" operator. Besides, this check showcases that since there are some totally subjective checks, there should also be checks to check them.

This chapter consists of all the ideas to be implemented in this work, but some of them may not be applicable within the scope of mvdXML. Also, based on the results of the checks, the subjective checks are still to be seen, whether they make sense or are useless. The implementation and results chapters discover and explain all those considerations thoroughly.

4.5 Implementation

The mvdXML standard, version 1.1, is the basis for this implementation. A thorough description of its specification can be found in Chapter 2.4. It is also recommended to read the documentation for more details (Chipman et al., 2016). This work is intended to know what is required to check a data model other than IFC using mvdXML by implementing a prototype of checking mvdXML using mvdXML. This chapter explains all the implementation steps successively, from implementing the furniture example and the mvdXML checks to executing the checker and testing.

4.5.1 Furniture Schema and Dataset

The following steps illustrated in Figure 4.9 are the main implementation steps to have several furniture mvdXML files that will help us showcase different combinations of attribute types and checking scenarios. It is important to mention that this example can be tailored to any other elements, i.e., it is not specific to the furniture and chairs example. The general framework is as follows: -

1. A general schema that has much flexibility
2. Data set based on this schema
3. MvdXMLs that check the data set
4. Incorrect mvdXML files

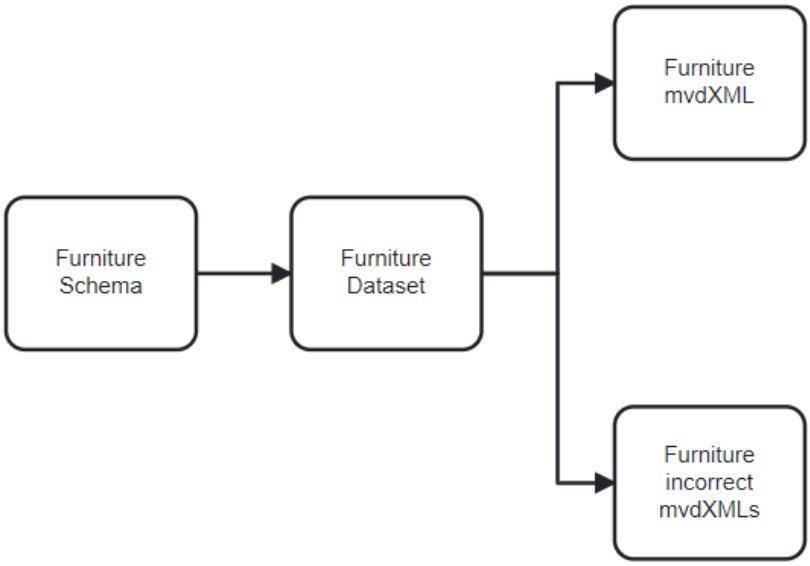

Figure 4.9: Framework of the furniture example

The first step in this example implementation is to tailor the furniture schema that has a lot of flexibility, with different types of attributes and their relationships. The UML diagram was introduced in Chapter 2 to explain schema and again outlined here in Figure 4.10 to describe the furniture schema in depth. In the schema, there are seven related classes. The schema begins with a furniture class that contains a chair class. The chair class is considered the main class in our dataset, with several mandatory and optional attributes of different types, such as the "Length" and "Width" attributes of type double and other boolean attributes. It also inherits the integer "ID", the string "Color" attributes, and another two objects, "Structure" and "Cushion" of type Material from another class, called component. the "Structure" and "Cushion" objects have two attributes: "Name" of type string and "Category" of type enumeration. In addition, the chair class has three classes: Leg, Armrest, and Production. Leg and armrest classes inherit the same attributes and objects of ID, Color, Structure, and Cushion from the component class. The leg class also has two other bool attributes, "HasWheel" and "IsSwiveling", while the Armrest class is empty. The production class includes different types of attributes as well. The XML schema definition (XSD) of the entire schema is attached in Appendix A.1.

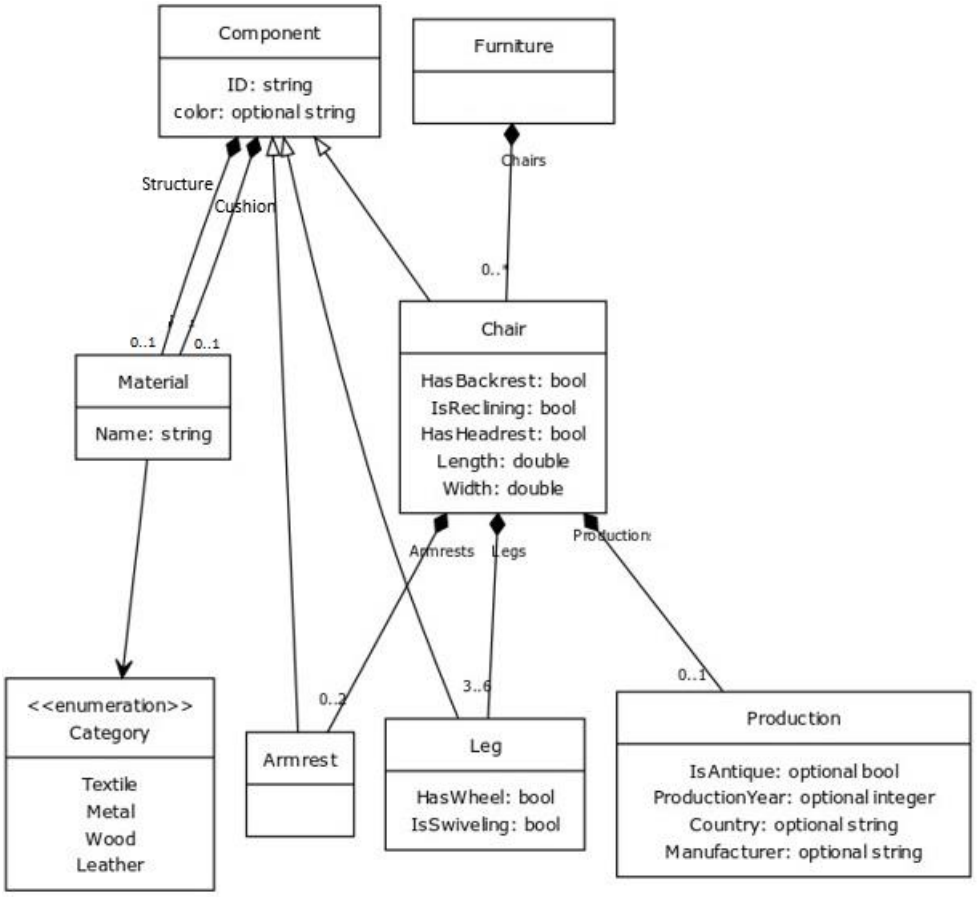

Figure 4.10: UML class diagram of the furniture schema

The next step is creating a furniture dataset based on the furniture schema. A furniture instance with four chair instances in the same XML file is designed. Each chair object stores a different collection of information. The UML diagram of the furniture instance, along with a detailed description of one chair object, Chair 3, is depicted in Figure 4.11 as an example. The characteristics of each chair object are specified so that each chair is unique to the others to give different results when checking it. For example, Chair 3 has all the required and optional attributes of ID, Color, Length, HasBackrest, HasHeadrest, and IsReclining, with specific values. The only optional attribute that was not stated is Width. Additionally, it has a Production instance from the Production class in which there is information for IsAntique, ProductionYear, and Country attributes but does not have the optional attribute Manufacturer. Chair 3 also has Structure and Cushion as attributes of the type material. The Category enumeration is metal and textile, while the Name is steel and silk, respectively. Moreover, Chair 3 is composed of three legs and one armrest. The three legs instances have all the attributes except Color. Two have the same characteristics, while the third has only one different characteristic: a wheel. The armrest does not contain information regarding the Color and

Structure attributes but only information on the Cushion attribute. The other three chairs follow the same concept as Chair 3. Appendix A.2 shows the full XML created, consisting of the four chair instances.

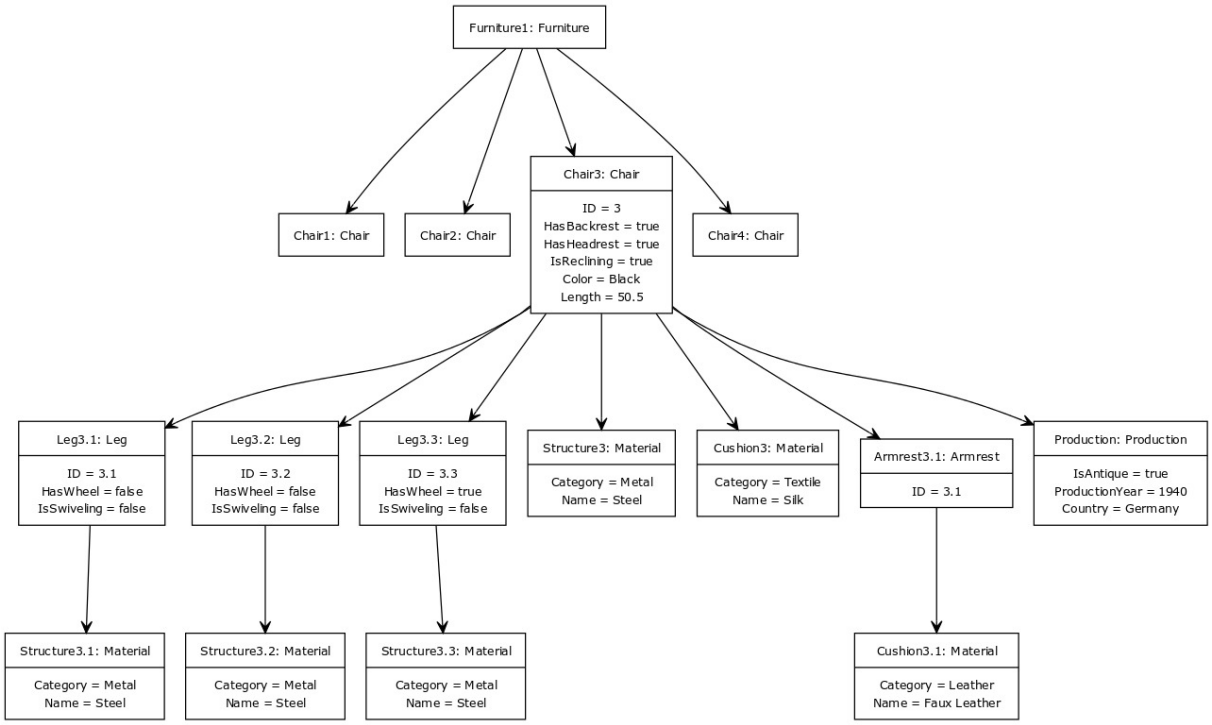

Figure 4.11: UML object diagram of chair3

The following step is constructing a Furniture mvdXML to check some features of the Furniture dataset elements. It consists of four concept templates referring to Component, Chair, Armrest, and Leg entities and nine concept roots representing nine different checks. The main concept template in our mvdXML is the Chair concept template, which refers to another two partial concept templates for the classes' Armrest and Leg. The two ways to refer to another entity and its attribute in a concept template is showcased in the Furniture mvdXML file. The first way is shown in the case of the Chair referring to the Armrest and Leg partial concept templates. To showcase the second way of the tree structure of attributes and entities, the Material class is included in the three concept templates using the tree structure of AttributeRule, called Structure or Cushion, referring to EntityRule Material, referring to its attributes. Both techniques are shown below in Figure 4.12. Finally, a separate concept template for the component is created to visualize some cases discussed below.

```
<ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
  <Definitions>
    <Definition>
      <Body />
    </Definition>
  </Definitions>
  <Rules>
    <AttributeRule AttributeName="ID" RuleID="ID" />
    <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
    <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
    <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
    <AttributeRule AttributeName="Color" RuleID="Color" />
    <AttributeRule AttributeName="Length" RuleID="Length" />
    <AttributeRule AttributeName="Width" RuleID="Width" />
    <AttributeRule AttributeName="Structure" RuleID="Structure">
      <EntityRules>
        <EntityRule EntityName="Material">
          <AttributeRules>
            <AttributeRule AttributeName="Name" RuleID="StructureMaterialName" />
            <AttributeRule AttributeName="Category" RuleID="StructureCategory" />
          </AttributeRules>
        </EntityRule>
      </EntityRules>
    </AttributeRule>
    <AttributeRule AttributeName="Cushion" RuleID="Cushion">
      <EntityRules>
        <EntityRule EntityName="Material">
          <AttributeRules>
            <AttributeRule AttributeName="Name" RuleID="CushionMaterialName" />
            <AttributeRule AttributeName="Category" RuleID="CushionCategory" />
          </AttributeRules>
        </EntityRule>
      </EntityRules>
    </AttributeRule>
    <AttributeRule AttributeName="Armrests" RuleID="Armrests">
      <EntityRules>
        <EntityRule EntityName="Armrest">
          <References IdPrefix="Armrests_">
            <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
          </References>
```

Figure 4.12: Part of the furniture mvdXML shows the two ways of referring to another entity

The following checks are defined specifically for this example which are solely based on the subjective point of view of the author. All the checks and their description are outlined in Table 4.4. Nine checks are implemented in the furniture mvdXML to check the dataset for several different features. It is interesting to view how the checker responds to some checks, such as checks 3,5,7,8 and 9. In check 5, we are looking for the existence of the structure attribute if the cushion attribute exists for any component element. This check ensures that the checker responds well to such a check, in which it will not only check one element but three; chair, leg, and armrest since they all inherit these attributes from the component. Other checks, 3 and 7, have Armrest and Chair as the applicableRootEntity in the ConceptRoot but refer to the component ConceptTemplate to check the values of the attributes of structure and cushion because there are inheritance relationships between them. All those checks are expected to work well since they are not different from the concept of IFC checks, except for check 8 and 9. Regarding checks 8 and 9, a new feature must be supported within the mvdXML scope: check two RuleIDs against each other rather than checking an attribute against a specific value. Hence, the checker is challenged with a new way of checking. Since it is an example that we are confident of its content, the results of such checks are already known. Chapter 5 discusses the results in depth.

Table 4.4: List of the furniture mvdXML checks. Green-colored checks are of high interest, while red-colored checks expected to be more challenging in implementation

Furniture mvdXML Checks	
Check ID	**Check Description**
1	A chair reclines, has a headrest, and at least one armrest if it has a backrest
2	If a chair is antique, then the production year, country, and the manufacturer must be known
3	If the structure material of a chair is steel, then the cushion material must be faux leather
4	The length and width of a chair must be equal to or greater than 52.5 and 48.5, respectively
5	If cushion attribute exists, then the structure must exist for the chair, armrest, and leg
6	A leg must swivel if it has a wheel
7	If the structure category for an armrest is metal, then it must be steel
8	The cushion materials of the chair and its armrests are similar
9	The colors of a chair and its legs are similar

Table 4.5 outlines a summary of the checks' concept templates and concept roots. The full individual mvdXML files can be found attached in Appendix A.3.

Table 4.5: Definition of the checks' concept templates and concept roots. Green-colored checks are of high interest, while red-colored checks expected to be more challenging in implementation

Furniture mvdXML implementation				
Check ID	**Rule Implementation**			
	ConceptTemplate	ConceptRoot		
	applicableEntity	applicableRootEntity	Applicability	TemplateRule
1	Chair	Chair	HasBackrest exists	IsReclining & HasHeadrest exist Armrest size > 0
2	Chair	Chair	IsAntique exists	Country, Manufacturer, & ProductionYear exist
3	Component	Chair	StructureMaterial Name=Steel	CushionMaterialName=Faux Leather
4	Chair	Chair	----------	Length>=52.5 Width>=48.5
5	Component	Component	Cushion exists	Structure exists
6	Leg	Leg	HasWheel exists	IsSwiveling exists
7	Component	Armrest	StructureCategory=Metal	StructureMaterialName=Steel
8	Chair	Chair	Armrest size > 0	CushionMaterialName=Armrests_CushionMaterial-Name
9	Chair	Chair	----------	Color=Legs_Color

Finally, for each defined mvdXML check, an mvdXML file, based on the furniture schema, consisting of one or several furniture checks with intended errors, is created to be captured by the developed checker. This step is essential to test the mvdXML checks and ensure the ability of the checker to catch several types of errors. Twelve mvdXML files are developed, including different types of errors. They are summarized in Table 4.6, while the complete mvdXML files can be found in Appendix A.4. Regarding EntityRule check, there is no need to create specific incorrect mvdXML for it because there are already several attributes without entities in the furniture mvdXML.

Table 4.6: List of furniture incorrect mvdXML files

Incorrect mvdXML files		
Name	**Error**	**Check**
ComponentRuleIDUnique-ness	RuleID (MaterialName) repeated twice within the same scope	RuleID
ComponentRuleIDExist-ence	RuleID does not exist for cushion, structure, name, and category attributes	
ChairIDPrefixUniqeness	Armrest and leg have the same IdPrefix	IdPrefix
ChairIDPrefixExistence	Armrest and leg do not have IdPrefix	
ComponentMetricValue	There is no MetricValue or incorrect one	MetricValue
ComponentParametersAt-tributes	There is no RuleID called StructureName or Cate-gory	ParametersAttributes
ChairTemplateReference	No template is referred to in the Applicability, Con-cept, and EntityRule nodes	TemplateReference
Chair-LegEntityReference	Leg ConceptRoot refers to Chair ConceptTem-plate	EntityReference

Table 4.6 List of furniture incorrect mvdXML files, continued

ChairReferencedConceptTemplates	EntityRule, ConceptRoot, and Concept refer to concept templates that does not exist in the mvdXML	ReferencedConceptTemplates
ChairConstraints	Existence of empty parameters in Applicability and Concept nodes, and no constraints exist	Constraints
ChairName	Name attribute is empty for mvdXML, ConceptTemplate, ModelView, ConceptRoot, and Concept	Name
ChairOperatorExistence	No operator is specified while there is more than one TemplateRule	OperatorExistence

4.5.2 MvdXML Checks

The main point to remember again while implementing mvdXML checks is that the main focus is not checking if the file is valid according to the schema or not since there are already available tools that could check schema compliance. This work checks certain optional features and criteria to be fulfilled. Added to this note, while implementing the Applicability in the ConceptRoot, there is no need to consider the existence of an element if checking its value, size, uniqueness, or type since the existence is already implicitly checked before the main check itself. For example, in the RuleID uniqueness check, we only implement "RuleID[Unique]=TRUE" in the parameters string without any applicability condition. If RuleID does not even exist, then this entity fails.

Each check is created in a separate file at the beginning of the implementation in order to be able to check each one independently on the furniture mvdXML first to track any source of errors easily. Then all the applicable checks, which proved their usefulness, are merged in one mvdXML file, called mvdXMLChecks, which is used to check the official mvdXMLs and itself. In total, this paper presents eleven mvdXML files with

different checks. As shown in Table 4.7, the checks use all metrics except for the metric value "type". In addition, Regex expression is employed in a significant check, such as MetricValue. All those checks with these metric values and Regex combinations are expected to be easily applied, as their logic and implementation are similar to IFC checks. Other different checks can be a little bit challenging or maybe even totally inapplicable. Such examples are ParametersAttributes, EntityReference, and ApplicableRootEntity. The first check tries to use a RuleID within a Regex expression, while the other two checks apply a two-sided check within different concept templates. All the mvdXML checks files are available in Appendix B.

Table 4.7: ConceptTemplate(s) and ConceptRoot(s) of the proposed mvdXML checks. The red-colored checks indicate a challenge in the implementation

MvdXML implementation					
ID	MvdXML	Rule Implementation			
		ConceptTemplate	ConceptRoot		
		applicableEntity	applicableRootEntity	Applicability	TemplateRule
1a	RuleID	ConceptTemplate	ConceptTemplate	-------	RuleID unique
1b		AttributeRule	AttributeRule	-------	RuleID exists
2a	IdPrefix	ConceptTemplate	ConceptTemplate	IdPrefix exists	IdPrefix is unique
2b		EntityRule	EntityRule	References exists	IdPrefix exists
3	MetricValue	TemplateRulesTemplateRule	TemplateRulesTemplateRule	-------	Using Regex to ensure Parameters contain an applicable metric
4	ParametersAttributes	mvdXML	mvdXML	-------	Parameters [Value]=reg'.*(RuleID[Value]).*'
5	TemplateReference	ConceptRoot	ConceptRoot	Applicability exists	@ref != ‘ ’
		Concept	Concept	-------	
		EntityRule	EntityRule	References exists	

Table 4.7 ConceptTemplate(s) and ConceptRoot(s) of the proposed mvdXML checks. The red-colored checks indicate a challenge in the implementation, continued

6	EntityReference	mvdXML	mvdXML	-------	ref=uuid AND EntityName=applicableEntity
7a	ReferencedConceptTemplates	mvdXML	mvdXML	Applicability exists	Applicabilityref=uuid
7b				-------	Conceptref=uuid
7c				References exists	Entityref=uuid
8	Constraints	mvdXML	mvdXML	Parameters=""	Constrain exists
9	Name	mvdXML	mvdXML	-------	Name exists
		ConceptTemplate	ConceptTemplate		
		ModelView	ModelView		
		ConceptRoot	ConceptRoot		
		Concept	Concept		
10	EntityRule	AttributeRule	AttributeRule	-------	EntityRule exists
11	OperatorExistence	ConceptRoot	ConceptRoot	A_TemplateRule size>1	A_Operator exists
		Concept	Concept	C_TemplateRule size>1	C_Operator exists

4.5.3 Checker

The checker was provided to me by the company The Hard Code GmbH, which I further developed and built upon it by implementing the necessary interfaces to fit with the furniture schema first and then to fit with checking mvdXML with mvdXML. Next, test units are developed for each stage of the testing process to test the procedure thoroughly. Each unit test needs two inputs, the dataset, and the checking file, giving out one output, the markdown report. The checker works so that schema-level checks are implemented before executing the mvdXML check. If there is a schema-level error, it throws an exception and does not execute the test. Hence, no need to design specific checks based on the schema. It supports the five primary checks of value, size, type, existence, and uniqueness. The checker is further developed it also to check for two-sided rules.

As mentioned before, the Markdown report has been chosen instead of BCF because only a pass/fail result is required. BCF has much more to do than report a pass or fail result, while a markdown report is much simpler. Thus, the checker itself generally works with the same logic as the IFC checker, but instead of IFC and BCF, mvdXML and Markdown. The report's structure, as shown in Figure 4.13, is straightforward, showing the report's name first, then the checked entity, followed by the ConceptRoot in which the check rules are enclosed. The check results are then given by showing how many elements of this entity are applicable and how many are not. The applicable ones are then checked to know how many of those are failing and how many are passing. Even though this may sound redundant to show both the failed and passed entities, it offers confidence that the checker works well in each detail.

```
# (Report Name)

## (applicableRootEntity)

### ConceptRoot (name) (uuid)

There were (count) applicable and (count) not applicable (applicableRootEntity(s)) in the checked file.

The following entities are failing:
- applicableRootEntity: .........
- applicableRootEntity: .........
- applicableRootEntity: .........
.........

The following entities are passing:
- applicableRootEntity: .........
- applicableRootEntity: .........
- applicableRootEntity: .........
.........

```

Figure 4.13: Structure of the report

5 Testing and Results

This chapter discusses each testing phase done in this work in depth. The testing procedure uses different datasets for each stage. The results of each phase and its effect on the implementation procedure are reviewed. Then, a general analysis of the checks is outlined in Chapter 6, along with graphs to illustrate and visualize the results.

5.1 Furniture Dataset

As discussed before, the first step of the testing procedure is checking the furniture XML file, consisting of a collection of information about four different chairs, against the designed furniture mvdXML. A manual check of the furniture dataset against its mvdXML has already been performed, and its results are summarized in Table 5.1.

Table 5.1: Results of the manual check of the furniture dataset against the furniture mvdXML

Manual Check Results				
Check ID	**Chair ID**			
	1	**2**	**3**	**4**
1	Fail	Not applicable	Pass	Pass
2	Not applicable	Pass	Fail	Pass
3	Not applicable	Pass	Fail	Pass
4	Pass	Fail	Fail	Pass
5	Fail	Pass	Pass	Pass
8	Not applicable	Pass	Fail	Pass
9	Pass	Fail	Fail	Pass
	Legs			
5	Not applicable			
6	Pass	2.1, 2.2, 2.3, 2.4 Fail	3.1, 3.2 Not applicable 3.3 Fails	Pass
	Armrests			
5	-------	Pass	3.1 Fails	Pass
7	-------	2.1 Fails 2.2 Pass	Not applicable	Pass

Figures 5.1 and 5.2 illustrate the report results of the chair, leg, and armrest entities. It shows how many instances are applicable or not and which ones failed and passed by representing their ID attribute. The results of the above manual check are precisely the same as the results of the generated report of the automatic check.

```
# Furniture Dataset test results

## Chair

### ConceptRoot 'Chair HasBackrest, then IsReclining, has Armrests, HasHeadrest' (uuid=32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 1

The following entities are passing:
- Chair: 3
- Chair: 4

### ConceptRoot 'Chair Production' (uuid=0c7b70bc-d061-4ff7-bb93-daf67c428ac1)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair Structure=steel, then Cushion=faux leather' (uuid=86f2a7d8-8e90-485a-a700-f3a8d0f0693b)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair length and width' (uuid=be790918-5b48-416c-9dd2-ccbe1b51f86a)

The following entities are failing:
- Chair: 2
- Chair: 3

The following entities are passing:
- Chair: 1
- Chair: 4
### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 4 applicable and 0 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 1

The following entities are passing:
- Chair: 2
- Chair: 3
- Chair: 4

### ConceptRoot 'Chair and Armrests Cushion similar' (uuid=81e78ab9-5560-4ce3-a4d6-7cad75f8920b)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair and Leg colors similar' (uuid=6cc834bf-aced-4f7d-9e20-7c6fd09096da)

The following entities are failing:
- Chair: 2
- Chair: 3

The following entities are passing:
- Chair: 1
- Chair: 4

```

Figure 5.1: Report showing the results of chair entity

```
## Leg
### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 0 applicable and 13 not applicable  Leg(s) in the checked file.

### ConceptRoot 'Leg HasWheel, then IsSwiveling' (uuid=74884d5c-d231-4204-aaeb-3a3fbc13049d)

There were 11 applicable and 2 not applicable  Leg(s) in the checked file.

The following entities are failing:
- Leg: 2.1
- Leg: 2.2
- Leg: 2.3
- Leg: 2.4
- Leg: 3.3

The following entities are passing:
- Leg: 1.1
- Leg: 1.2
- Leg: 1.3
- Leg: 4.1
- Leg: 4.2
- Leg: 4.3

## Armrest
### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 4 applicable and 0 not applicable  Armrest(s) in the checked file.

The following entities are failing:
- Armrest: 3.1

The following entities are passing:
- Armrest: 2.1
- Armrest: 2.2
- Armrest: 4.1

### ConceptRoot 'Armrest StructureCategory is metal, then steel' (uuid=811c5e38-684d-4684-a54d-a5c4cc0c7517)

There were 3 applicable and 1 not applicable  Armrest(s) in the checked file.

The following entities are failing:
- Armrest: 2.1

The following entities are passing:
- Armrest: 2.2
- Armrest: 4.1
```

Figure 5.2: Report showing the results of leg and armrest entities

The second step of the evaluation procedure also includes the furniture schema, in which the furniture mvdXML file is checked against the mvdXML checks. Then the incorrect mvdXML files are checked against the mvdXML checks also.

First, several unit tests are implemented according to the number of checks. Each unit has furniture mvdXML as the first fixed input and an mvdXML check as the changing second input. Thus, eleven-unit tests are implemented and executed on the furniture mvdXML. The following matrix in Figure 5.3 classifies the reports into four categories: true positive, false positive, false negative, and true negative. True positive means that regarding this check on furniture mvdXML, some/all elements passed as expected. While false positive means that some/all elements passed as not expected. True and false negatives are the concept in which the elements fail. For example, EntityRule check gives pass and fail results for different AttributeRule(s) in the furniture mvdXML file, which is true as expected since some have EntityRule, and others do not.

	True	False
Positive	**True Positive (TP)** RuleID IdPrefix MetricValue TemplateReference ReferencedConceptTemplates Constraints Name EntityRule OperatorExistence	**False Positive (FP)** ----------
Negative	**False Negative (FN)** ParametersAttributes EntityReference ReferencedConceptTemplates – EntityConceptRoot	**True Negative (TN)** EntityRule

Figure 5.3: Confusion matrix of mvdXML checks on the furniture mvdXML

Thus, the following graph in Figure 5.4 visualizes that from those eleven results, the results of two check does not make sense and are false, which are ParametersAttributes and EntityReference. Error sources of those checks are thoroughly discussed in Chapter 6. All the related eleven reports can be found in Appendix C.

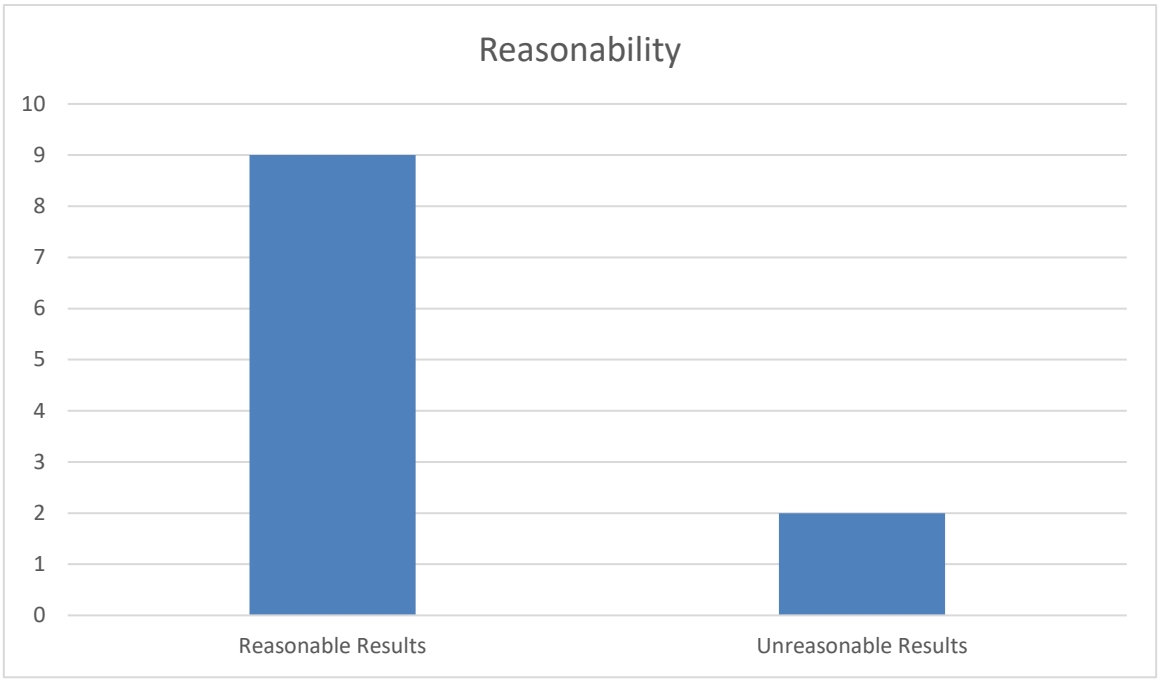

Figure 5.4: Histogram showing the number of reasonable and unreasonable results of the furniture mvdXML

Next is to check the designed incorrect furniture mvdXMLs against the mvdXML checks to view how many of the intended errors can be caught and compare the results with the above results. Twelve unit tests are created, each with the inputs of one mvdXML check and the respective incorrect mvdXML file. The findings from the developed unit tests are elaborated in Figures 5.5 and 5.6. Since TemplateReference and ReferencedConceptTemplates are schema checks, the checker was adapted so that it does not check the incorrect files against the schema first. Instead, it just executes the mvdXML check.

	True	False
Positive	**True Positive (TP)** ----------	**False Positive (FP)** ChairOperatorExistence
Negative	**False Negative (FN)** ----------	**True Negative (TN)** ComponentRuleIDUniqueness – RuleID ComponentRuleIDExistence - RuleID ChairIDPrefixExistence - IdPrefix ChairIDPrefixUniqueness – IdPrefix ComponentMetricValue – MetricValue ComponentParametersAttributes – ParametersAttributes ChairTemplateReference – TemplateReference Chair-LegEntityReference – EntityReference ChairReferencedConceptTemplates – ReferencedConceptTemplates ChairConstraints – Constraints ChairName - Name

Figure 5.5: Confusion matrix of mvdXML checks on the incorrect furniture mvdXMLs

Out of twelve errors, one file gives false positive results: OperatorExistence. On the other hand, opposite to the first checking case, we can view that ParametersAttributes, EntityReference, and the third ConceptRoot of ReferencedConceptTemplates give accurate results in this testing case. In contrast, they give false results in the first one. The following Figure 5.6 views the number of results that are reasonable and not reasonable. The next chapter compares and analyzes these findings to draw some essential conclusions. The detailed reports of the results are attached to Appendix C.

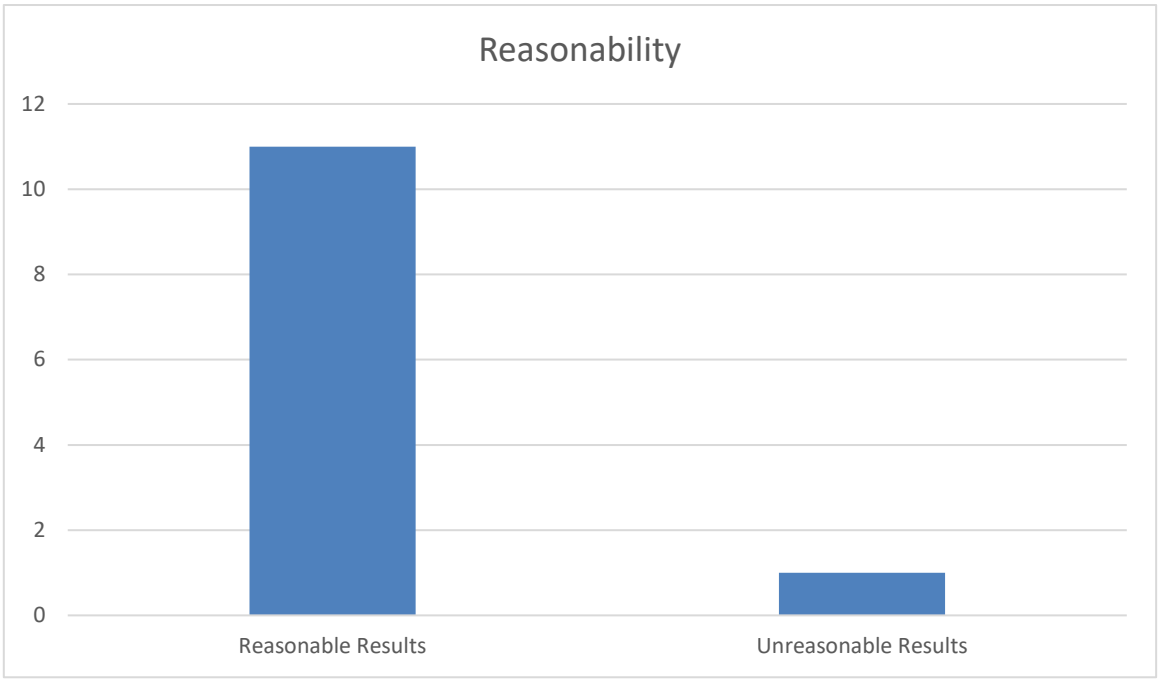

Figure 5.6: Histogram showing the number of reasonable and unreasonable results of the incorrect mvdXMLs

5.2 Other mvdXMLs

The third evaluation stage uses nine mvdXML files described in Section 4.3.3. The results show that out of the nine files, no mvdXML is 100% accurate and correct according to the subjective and objective checks. There are around five files, which contain several errors, either schema-level errors or errors based on the checks, while the other four have up to three errors. Table 5.2 recaps the files and their checking results.

Table 5.2: Summary of the discovered errors in the mvdXML files

MvdXML file	Error	MvdXML check
ReferenceView	Several AttributeRule(s) do not have RuleID	RuleID
	Several AttributeRule(s) do not have EntityRule	EntityRule
	Eight EntityRule(s) do not have IdPrefix	IdPrefix
	All TemplateRule(s) do not have metric value	MetricValue
	mvdXML and several ConceptTemplate(s) have empty @Name	Name
QuantityTakeOff	Several AttributeRule(s) do not have RuleID	RuleID
	Several AttributeRule(s) do not have EntityRule	EntityRule
	Six EntityRule(s) do not have IdPrefix	IdPrefix
	Several TemplatRule(s) do not have metric value	MetricValue
	MvdXML and several ConceptTemplate(s) have empty @Name	Name
IFC4precast	Several AttributeRule(s) do not have RuleID	RuleID
	Seven AttributeRule(s) do not have EntityRule	EntityRule
	MvdXML has empty @Name	Name

Table 5.2 Summary of the discovered errors in the mvdXML files, continued

KIT_Example1	AttributeRule: Name does not have EntityRule	EntityRule
	EntityRule:IfcSimpleProperty does not have IdPrefix	IdPrefix
KIT_Example2	Two AttributeRule(s) do not have RuleID	RuleID
	Several AttributeRule(s) do not have EntityRule	EntityRule
	Six EntityRule(s) do not have IdPrefix	IdPrefix
Documentation Example	Nine AttributeRule(s) do not have RuleID	RuleID
	Two AttributeRule(s) do not have EntityRule	EntityRule
	One ConceptRoot has empty @Name	Name
COBie	Several AttributeRule(s) do not have RuleID	RuleID
	Six AttributeRule(s) do not have EntityRule	EntityRule
	All TemplateRule(s) do not have metric value	MetricValue
	mvdXML, all ConceptRoot(s) and one Concept have empty @Name	Name
AlignmentBasedRV	Several AttributeRule(s) do not have RuleID	RuleID
	Several AttributeRule(s) do not have EntityRule	EntityRule
	Six EntityRule(s) do not have IdPrefix	IdPrefix
	Several TemplateRule(s) do not have metric value	MetricValue

Table 5.2 Summary of the discovered errors in the mvdXML files, continued

IFC4_GU	Several AttributeRule(s) do not have RuleID	RuleID
	Several AttributeRule(s) do not have EntityRule	EntityRule
	Six EntityRule(s) do not have IdPrefix	IdPrefix
	Several TemplatRule(s) do not have metric value	MetricValue
	Mvdxml has empty @Name	Name

Along with the above summary table of errors, several interesting errors and results are objectively highlighted in this section, while the next chapter discusses them thoroughly. The results of ReferenceView, for example, show that all TemplateRule(s) fail according to the MetricValue check because, as we can see in Figure 5.7, there is no metric value following the old grammar of mvdXML 1.0. There is also one TemplateRule that is empty. The same applies to COBie, QTO, and IFC4_GU mvdXMLs, which contain TemplateRule(s) that do not have metric values or are empty. Nevertheless, for the empty TemplateRule, as long as the Constraints check passes, then we can ignore it.

```
<TemplateRules operator="and">
  <TemplateRule Parameters="PsetName=Pset_ActuatorTypeCommon;" />
  <TemplateRule Parameters="PsetName=Pset_ActuatorTypeLinearActuation;" />
  <TemplateRule Parameters="PsetName=Pset_ActuatorTypeRotationalActuation;" />
</TemplateRules>
```

Figure 5.7: Part of the ReferenceView mvdXML showing its parameters element follow the old grammar of mvdXML 1.0

The checker checks all those files against a merged file of only the applicable and reasonable mvdXML checks based on the previous stages. Thus, ParametersAttributes, EntityReference, OperatorExistence, and the third ConceptRoot of ReferencedConceptTemplates checks were excluded. As mentioned before, the checker first validates the mvdXML file against the schema. Therefore, several adjustments had to be made on several files before executing the primary checks. For example, ReferenceView and QuantityTakeOff have been adapted since multiple of its ConceptTemplates did not have the required attribute @name. Hence, the attribute has been added but remained empty. All the failing ConceptTemplate entities, according to the @name

check, are the ones that did not have it in the first place. QTO mvdXML was also invalid according to the schema for having @status as mandatory, while status is an enumeration that does not have this option in mvdXML 1.1. It has been adapted, and then the checking procedure was accomplished.

Another mvdXML that was adjusted is COBie. All the empty @status attributes (status=" ") have been removed, followed by changing (status=" Candidate" to status=" candidate"). In addition, the attribute @name has been added to the mvdXML element. The value of the @applicablity attribute has also been changed from being "default" to "both" and the @requirement value from being "optional" to "recommended". Again, all those attributes are enumerations with specific options in mvdXML 1.1. Moreover, Figure 5.8 visualizes part of the COBie mvdXML report, in which one can visualize that "Rules" is encoded instead of "TemplateRules", which is again an old grammar of mvdXML 1.0. Hence, it has been edited to be used within the scope of this thesis.

```
<Rules>
  <TemplateRule Description="Height at lowest point from bottom of ceiling to top of flooring." Parameters="Type=IfcQuantityLength;Name=NetHeight;" />
  <TemplateRule Description="Area of floor within walls not including any obstructions such as columns." Parameters="Type=IfcQuantityArea;Name=GrossArea;" />
  <TemplateRule Description="Area of floor within walls with any obstructions subtracted." Parameters="Type=IfcQuantityArea;Name=NetArea;" />
</Rules>
```

Figure 5.8: Part of the COBie mvdXML

Furthermore, AlignmentBasedRV was adjusted so that each TemplateRule has the required attribute @Parameters, which was not the case. All the failed TemplateRule(s), according to the MetricValue check, were invalid according to the schema in the first place since there was already no @Parameters attribute.

5.3 MvdXML Checks

The last data model used in the procedure using the mvdXML checks is the mvdXML checks file itself. The same checks applied to it, and there are no errors found, which means the mvdXML is 100% accurate according to the checks. The report results in detail can be found in Appendix C.

6 Discussion

This chapter closely examines the results of all the employed testing phases. The results and findings are compared and illustrated to identify the accuracy variations and the applicability of the mvdXML checks. This discussion subjectively evaluates the results and the process.

In general, as mentioned throughout this paper, validation is a very subjective process. Hence, the whole implementation, testing, and evaluation procedures can be considered subjective, even if we tried to be as objective as possible. Overall, the checker proves its practicality and potential to check other data models and expand its usage. However, the method has some limitations that are also discussed in this chapter.

Figure 6.1 elaborates the objectiveness of the implemented checks. Downward are the objective checks, based on the mvdXML documentation, along with some schema checks, while upward are the subjective checks, based on the author's view. In between come RuleID and IdPrefix checks, which are Objective-Subjective since their existence is a subjective check, but together they have a logical objective reason behind them. In RuleID, the first ConceptRoot checks for the existence of RuleID to ensure that all referred attributes in Parameters have their RuleID, while the second ConceptRoot checks for the uniqueness of RuleIDs within only one ConceptTemplate. To check the uniqueness within multiple ConceptTemplates, IdPrefix checks were created to ensure the existence of a unique IdPrefix when referring to another ConceptTemplate. This flow would have been more effortless if the mvdXML 1.1 grammar enabled us to check further a specific entity based on the result of a predecessor check. So that one can specify that if the RuleID of two AttributeRules in different ConceptTemplate is not unique, then check if one of them has an EntityRule with a unique IdPrefix.

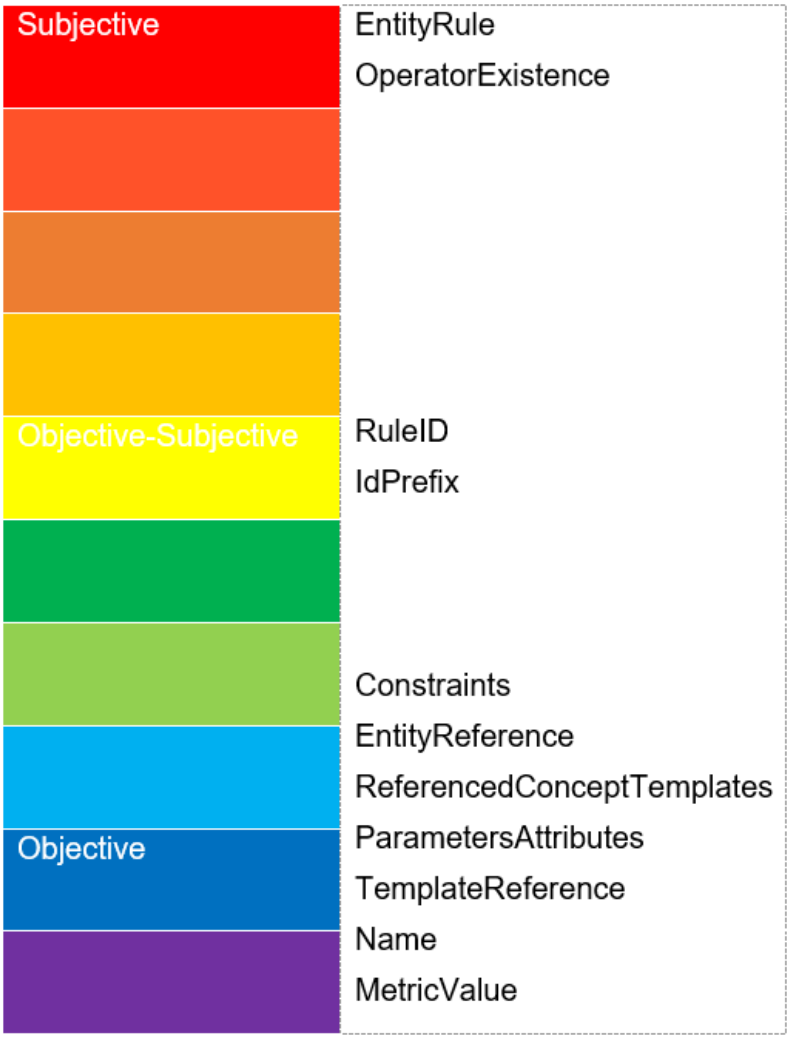

Figure 6.1: Scale of objectiveness of the mvdXML checks

From those eleven checks, two checks are inapplicable, EntityReference and ParametersAttributes, and thus give unreasonable results, while OperatorExistence is applicable and can be implemented but gives unreasonable results also. Therefore, around 73% of the checks give correct meaningful, valuable results. These findings can be visualized in Figure 6.2, followed by an explanation of the inapplicability and absurdity of the checks.

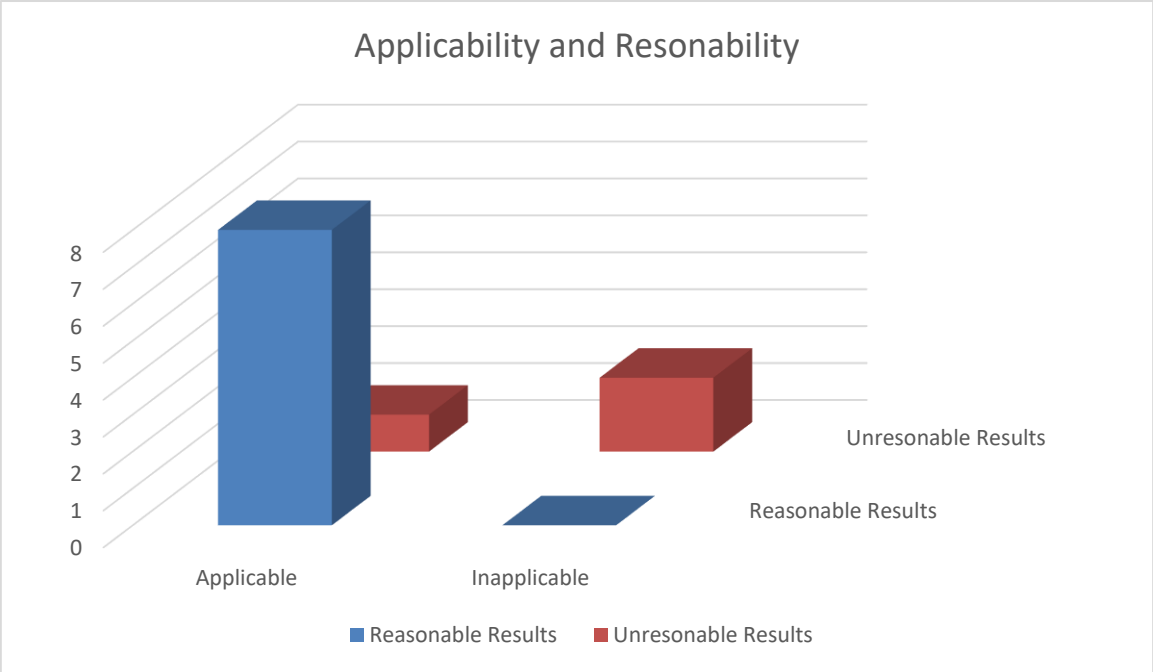

Figure 6.2: Chart showing the number of unreasonable and/or inapplicable checks

In EntityReference check, the concept of right-hand-sided RuleIDs did not work correctly because it only compares the @ref of the EntityRule against the @uuid of its ConceptTemplate, which in return always gives a "fail" result. After all, these two elements will never be equal. For example, as shown in Figure 6.3, when EntityReference check was executed on the furniture mvdXML, it showed a false negative result because it only compared the Armrest @ref with the first ConceptTemplate @uuid. In another check, such as ReferencedConceptTemplate, the right-hand-sided work perfectly because the two elements (RuleID) checked against each other are not within the same node, such as EntityReference check. In addition, the logic behind the EntityReference check has another limitation. The check is not only to check the @ref against @uuid but also to ensure that the EntityName is the same as the applicableEntity of the referred ConceptTemplate, which is challenging to implement.

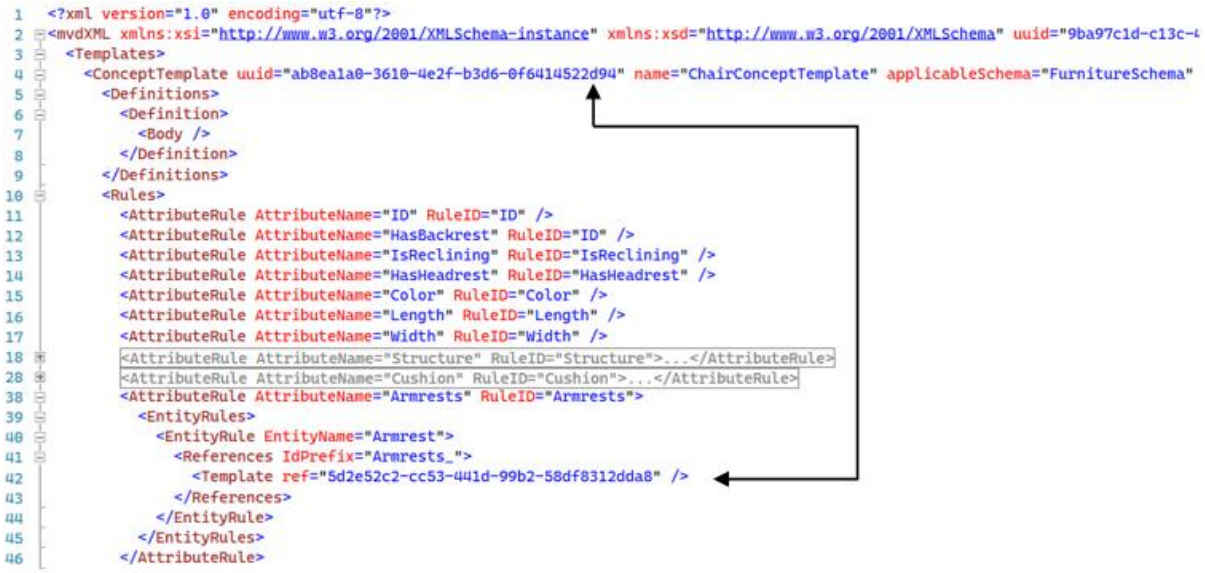

Figure 6.3: Part of the furniture mvdXML file showing how EntityReference check works

Furthermore, In the ParametersAttributes check, the rule tries to use a RuleID within a Regex expression (Parameters [Value]=reg'.*(RuleID[Value]).*'). Although this is a very significant objective check, it was inapplicable with the current grammar of mvdXML 1.1. Hence, an error such as that in ReferenceView official mvdXML, where there are no RuleID called “Spatial Composite” or “Spatial Parts” but mentioned in the Parameters string, was not caught due to the inapplicability of this check.

Regarding the OperatorExistence check, the rule syntax was easy to implement, but since there is always a default operator "and" to TemplateRules, the result is always "pass.” Therefore, there is no need for this check currently. Even though, subjectively saying, it is better in the future to remove the mandatory default "and" operator and to consider this check to allow the user to specify which specific operator he needs. Otherwise, if forgotten, the mvdXML file will consider it "and,” while the user may mean another logical operator.

To sum up, eight of eleven checks show applicability as well as meaningful results. The applicable eight checks proved their usefulness and can be used for future studies. Hence, they are enough to consider this prototypical implementation a successful methodology. A scale of the toughness of the implementation of those checks is shown in Figure 6.4. The figure illustrates how easy or complicated the implementation of such checks is within the scope of mvdXML grammar. As shown, ParametersAttributes and EntityReference are inapplicable and give wrong results. At the same time, ReferencedConceptTemplates is relatively difficult to implement since a new way of

supporting checks that have RuleIDs on both sides is developed, i.e., comparing two values from the same mvdXML file. This check has three concept roots for three different entities: Applicability, Concept, and EntityRule. The first two concept roots work correctly, while the third one, which is related to the EntityRule @ref, gives unmeaningful results, because same as EntityReference, it is trying to compare two RuleIDs within the same ConceptTemplate as shown in Figure 6.3. All other checks were applicable because they were implemented similarly to the usual concept of IFC checks.

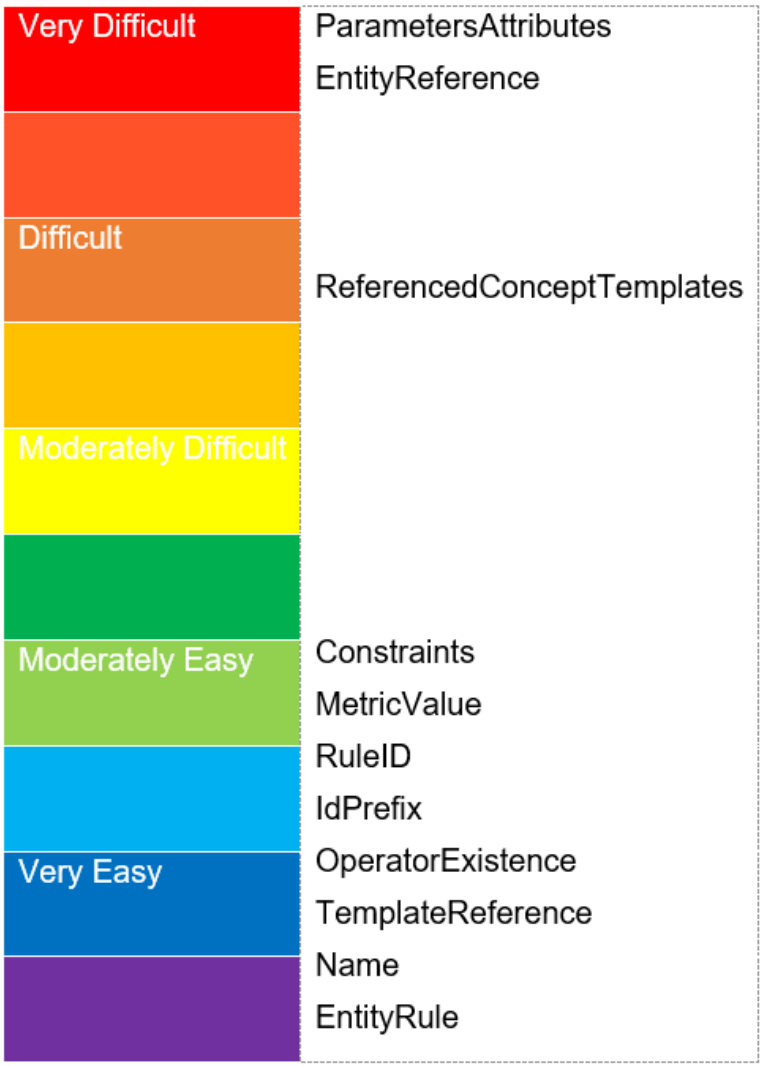

Figure 6.4: Scale of difficulty of the mvdXML checks

Similar to the above scale, Figure 6.5 visualizes another scale for evaluating the knowledge we get from each check. The scale shows that three checks are totally useless since they give false results. Next, come ReferencedConceptTemplates and Constraints checks which do not provide complete information about which entity is exactly failing since the applicableRootEntity is the root element mvdXML. This is because mvdXML 1.1 does not have the feature of referring to two ConceptTemplates,

in contrast to mvdXML 1.2. Additionally, the mvdXML schema currently defines the applicableRootEntity as a string, not an array of strings. The user cannot specify different entities to be checked in the right-hand-side rules. Hence, the root entity was set to mvdXML to be able to check two RuleIDs against each other from two different nodes. Therefore, the results are not very detailed and clarified due to the generality of the entity, i.e., the user knows this mvdXML file fails due to an error according to this check, but he must manually search for which exact entity caused this error. Moreover, in the Constraints check, the rule only ensures that there is a Constraint attribute in the whole mvdXML file if there is an empty Parameters string instead of explicitly checking the existence of this attribute within the specified referred ConceptTemplate of the blank Parameters string. Therefore, the user must continue manually for more information on this result. Finally, it can also be seen that the other six checks give 100% valuable results.

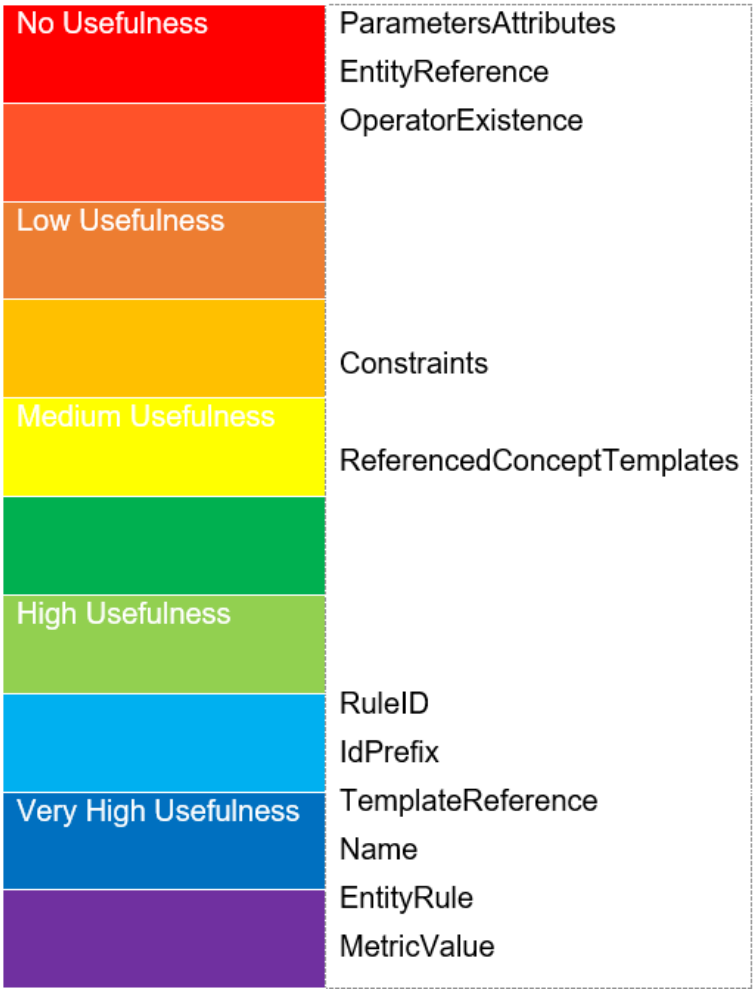

Figure 6.5: Scale of knowledge of the mvdXML checks

The pie chart in Figure 6.6 classifies the accuracy and correctness of the mvdXML files being tested into four categories, 100%, 75%, 50%, and 25% of correctness. This classification is based on two factors; the number of failed results and the number and severity of schema errors discovered. These results conclude that checking any mvdXML file, whether the official BuildingSMART ones or others created by other end users, is crucial. Furthermore, it can be observed that the bigger the size of the mvdXML file, the more prone it is to errors, whether errors according to the schema or to the developed checks. IFC4precast and IFC4 general usage mvdXMLs are the only two official mvdXMLs that did not have any schema error in the first place. Nevertheless, IFC4 general usage has five types of errors according to the mvdXML checks, while IFC4precast has only three. Along with IFC4precast, the mvdXML examples of KIT and the documentation show the highest accuracy after the mvdXML file of checks itself. The mvdXML checks file has 100% accuracy because it matches all the defined rules.

Generally, the mvdXML files from sources other than the official buildingSMART have more accuracy. This can mean the more complex the file is, the more its fragility towards mistakes. In the scenario of empty parameters but constraints check pass, the errors are ignored because there is a reason behind the parameter being blank. However, in the case of the absence of metric values within parameters, such as in ReferenceView, AlignmentBasedRV, and IFC4_GU, the errors were considered because it does not make sense that the expressions still follow the old style of the previous version, even if still supported. It must be encoded with the current grammar of mvdXML 1.1, which is more self-explanatory for both humans and computers.

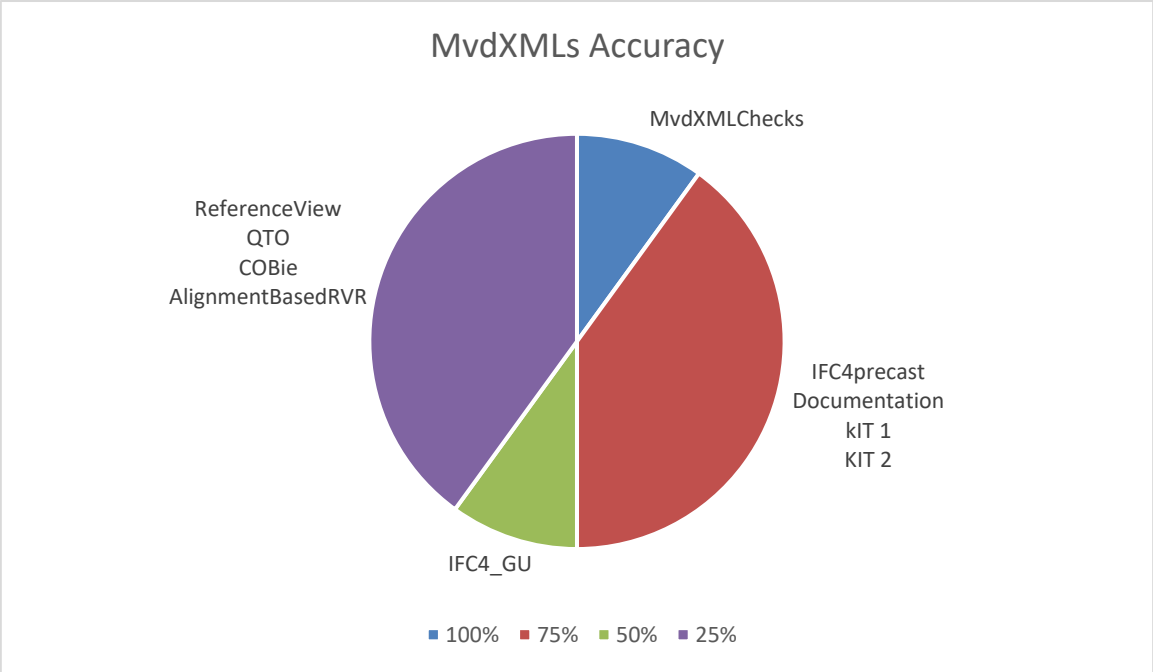

Figure 6.6: Pie chart mapping the accuracy of the mvdXML files according to the checks

To conclude, the prototypical implementation in this study has proved its success and usefulness. Using mvdXML to check itself can help guarantee the correctness of mvdXMLs, which is highly needed, as elaborated in the previous paragraphs. It thus has the potential also to be used on other data models. However, some limitations regarding mvdXML 1.1 grammar and schema must be addressed. The following bullet points sum up some of the discovered limitations: -

1. Support of rules with Right-hand sided RuleIDs within the same entity/node
2. Use of RuleID within a Regex expression
3. Referring to two ConceptTemplates in ConceptRoot
4. Setting different entities for Applicability and Concept within the same ConceptRoot
5. Selecting more than one entity in the ApplicableRootEntity
6. Selecting a specific entity with a particular value in the ConceptRoot to be checked

7 Summary

7.1 Conclusion

This thesis evaluated and analyzed the capabilities of the employment of mvdXML as a validation tool on data models other than IFCs by developing a checker model that checks mvdXML data models using mvdXML. The structure of this paper first introduced the concept of validation and verification, as well as the current state of mvdXML and its schema structure. Then it has been decided to validate mvdXML files using itself subjectively, and thus the correctness of mvdXML files has also been checked according to the designed checks. The checker used different datasets to check, including a simple furniture example designated by the author, along with other official BuildingSMART mvdXMLs and mvdXMLs from other sources. The results outcome has then been compared and analyzed to understand how mvdXML can support different types of checks and react to them.

After the first evaluation procedure on the checks using the furniture example, eight checks that performed the best within the scope of mvdXML were then merged into one mvdXML file called mvdXMLChecks. This mvdXML file can be used as a final version to check any mvdXMLs. The final chosen checks are RuleID, IdPrefix, TemplateReference, MetricValue, Constraints, Name, ReferencedConceptTemplates, and EntityRule. These rules check well and properly some features which are of use to our specified objectives, such as detecting missing elements, ensuring correct content within strings and correct values of attributes, as well as detecting mistakes in elements usage. Therefore, mvdXML proves that it can be further developed to be a more versatile tool. The end-user has the freedom to consider more or less of the developed checks in this work and to edit or update them to adapt them to his own use case or maybe also to develop the method further to support other data models.

In addition to the discovered capabilities of mvdXML, this implemented method has also discovered some limitations that were addressed in detail in the previous chapter, which concludes to the fact that the mvdXML needs to be expanded to show more flexibility to support more useful checks. It has been proven scientifically through this thesis that the schema needs more improvement and expansion in future versions.

7.2 Future Work

There are some limitations in this study and opportunities for further research. One major limitation of this study is that it focused on one version of mvdXML 1.1, which does not show all the new features that are supported by mvdXML 1.2. Further research could expand the implemented method to include more checks based on the latest 1.2 version to increase the checker efficiency. Additionally, the study looked only at the effects of the implementation of mvdXML checking mvdXMLs but omitted other data models. Further research could consider using mvdXML to validate other data models within the same field and scope to study how adaptable and flexible mvdXML is as a validation tool.

Furthermore, various limitations have been discovered in this thesis. Several recommendations for expanding the future schema that this or the past work have not covered can be introduced. It might be interesting for other researchers to develop further the schema and the checks based on it. These topics can encourage others to keep adding new features, coming up with fresh concepts and making improvements and enhancements.

- **Add a feature to use a RuleID within a Regex expression**

  In the current checks, we were not able to implement a check such as that in ParametersAttributes mvdXML (Parameters [Value]=reg'.*(RuleID[Value]).*'). In the future, the mvdXML schema can be expanded to enable the mvdXML to use RuleID within Regex expression, which will help develop more checks and enhance their quality.

- **Enable Right-hand sided RuleIDs within the same node**

  In the future, the Right-hand sided RuleIDs rules should be improved to work correctly within the same element. For EntityReference check, for example, we need to add the support of a feature to check each @ref attribute with all the other ConceptTemplate @uuids, as shown in Figure 7.1. The same concept follows for the third ConceptRoot of ReferencedConceptTemplates check.

Figure 7.1: Illustration showing how right-hand-side RuleID within the same node should work

- **Enable referring to a particular entity with a specific feature**
  The schema should be expanded to allow checking particular entities with specific values. In other words, regarding the EntityReference check, the user should be able to choose the referred ConceptTemplate in the EntityRule node and apply another check on it. Figure 7.2 clarifies this check concept, which is currently inapplicable.

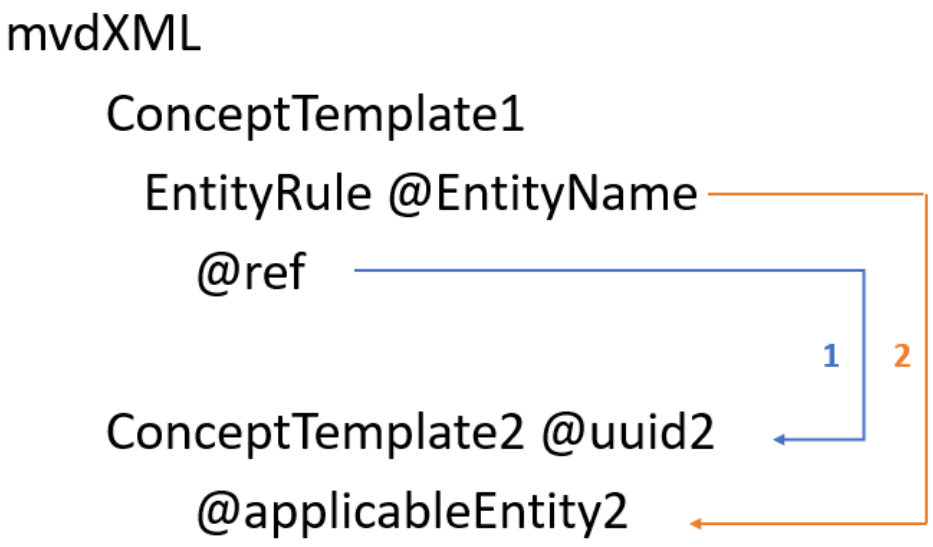

Figure 7.2: Illustration showing how EntityReference check should work

- **Update the checks based on the grammar of mvdXML 1.2 + new features**
  MvdXML 1.2 supports referring multiple ConceptTemplates since it was a requirement that came up between 1.1 and 1.2. Since needs are constantly expanding, an additional provision that would support and enhance this feature is to set applicableRootEntity as an array of strings to enable the user to specify two entities, one primary entity and another secondary entity. This enhancement would again help checks such as the Constraints check to be more self-explanatory.
- **Add totally new checks**
  We have already implemented this set of rules based on some defined objectives from the author's point of view. The checker can also have new checks to fulfill other goals and purposes. Moreover, a new checking procedure with an additional set of checks can be implemented to check the developed checks in this paper. It would be somehow an infinite process of reviewing the checks since any checker is prone to errors, and thus it would be very wise to define another set of rules which checks that the checker's checks make sense. For example, the OperatorExistence check was easily implemented but generated non-sense results. Such rules must be reviewed to guarantee the perfect quality of the whole procedure for generating mvdXMLs.

References

Azhar, S. (2011). Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and management in engineering*, *11*(3), 241-252.

Baumgärtel, K., Pirnbaum, S., Pruvost, H., & Scherer, R. J. (2016). Automatic BIM filtering using model view definitions. 33rd CIB W78 conference, Brisbane, Australia,

Bolchini, C., & Salice, F. (2000). The design of self-checking systems. Proc. 1st On-Line Symposium for Electronics Engineerings (OSEE),

Booch, G. (2005). *The unified modeling language user guide*. Pearson Education India.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2015). *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. Springer-Verlag.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2018). Building information modeling: Why? what? how? In *Building information modeling* (pp. 1-24). Springer.

buildingSMART. IFC Formats. Retrieved: https://technical.buildingsmart.org/ standards/ifc/ifc-formats/. Accessed: 30.10.2022

buildingSMART. Industry Foundation Classes (IFC) – An Introduction. Retrieved: https://technical.buildingsmart.org/ standards/ifc/. Accessed: 30.10.2022

buildingSMART. Model View Definition (MVD) – An Introduction. Retrieved: https://technical.buildingsmart.org/ standards/ifc/mvd/. Accessed: 30.10.2022

buildingSMART. The curious case of the MVD. Retrieved: https://blog.buildingsmart.org/blog/the-curious-case-of-the-mvd/ . Accessed: 30.10.2022

buildingSMART. mvdXML. Retrieved: https://technical.buildingsmart.org/ standards/ifc/imvd/mvdxml/. Accessed: 30.10.2022

buildingSMART. MVD Database. Retrieved: https://technical.buildingsmart.org/ standards/ifc/imvd/mvd-database/. Accessed: 30.10.2022

buildingSMART. Problems when parsing and automating ReferenceView_V1-2.mvdxml. Retrieved: https://forums.buildingsmart.org/t/problems-when-parsing-and-automating-referenceview-v1-2-mvdxml/1935. Accessed: 30.10.2022

Cerovsek, T. (2011). A review and outlook for a 'Building Information Model'(BIM): A multi-standpoint framework for technological development. *Advanced engineering informatics*, *25*(2), 224-244.

Chipman, T., Liebich, T., & Weise, M. (2012). mvdXML. In: Retrieved from: Specification of a standardized format to define and ….

Chipman, T., Liebich, T., & Weise, M. (2016). mvdXMl specification 1.1. *Model Support Group (MSG) of buildingSMART International Ltd*.

Cunningham, T. (2013). Factors affecting the cost of building work-an overview.

Di Zio, M., Fursova, N., Gelsema, T., Gießing, S., Guarnera, U., Petrauskienė, J., Quensel-von Kalben, L., Scanu, M., ten Bosch, K., & van der Loo, M. (2016). Methodology for data validation 1.0. *Essnet Validat Foundation*.

Dykes, L., & Tittel, E. (2011). *XML for Dummies*. John Wiley & Sons.

EN 17412-1:2020. (2020). Building Information Modelling - Level of Information Need.

Gao, J., Xie, C., & Tao, C. (2016). Big data validation and quality assurance--issuses, challenges, and needs. 2016 IEEE symposium on service-oriented system engineering (SOSE),

Hietanen, J., & Final, S. (2006). IFC model view definition format. *International Alliance for Interoperability*, 1-29.

Hjelseth, E. (2016). Classification of BIM-based model checking concepts.

ISO 8879:1986. (1986). Information processing — Text and office systems — Standard Generalized Markup Language (SGML).

ISO 29481-1:2010. (2010). Building information modelling — Information delivery manual — Part 1: Methodology and format.

ISO 9000-3:2015. (2015). Quality management systems — Fundamentals and vocabulary.

Jaud, Š., & Muhič, S. (2022). Checking IFC with MVD Rules in Infrastructure: A Case Study. *Engineering Proceedings*, *17*(1), 33.

Jiang, S., Wu, Z., Zhang, B., & Cha, H. S. (2019). Combined MvdXML and semantic technologies for green construction code checking. *Applied Sciences*, *9*(7), 1463.

Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology (KIT) (2021). KIT mvdXML Examples. Retrieved: https://www.ifcwiki.org/index.php?title=KIT_mvdXML_Examples

Laakso, M., & Kiviniemi, A. (2012). The IFC standard: A review of history, development, and standardization, information technology. *ITcon*, *17*(9), 134-161.

Lee, Y.-C., Eastman, C. M., & Solihin, W. (2016). An ontology-based approach for developing data exchange requirements and model views of building information modeling. *Advanced engineering informatics*, *30*(3), 354-367.

Liu, H., Gao, G., Zhang, H., Liu, Y.-S., Song, Y., & Gu, M. (2019). MVDLite: A Light-weight Representation of Model View Definition with Fast Validation for BIM Applications. *arXiv preprint arXiv:1909.06997*.

Mosleh, M. A., Alhussein, M. A., Baba, M. S., Malek, S., & ab Hamid, S. (2016). Reviewing and classification of software model checking tools. In *Advanced Computer and Communication Engineering Technology* (pp. 279-294). Springer.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, *24*(3), 45-77.

Sidi, F., Panahy, P. H. S., Affendey, L. S., Jabar, M. A., Ibrahim, H., & Mustapha, A. (2012). Data quality: A survey of data quality dimensions. 2012 International Conference on Information Retrieval & Knowledge Management,

Sun, J., Liu, Y., & Cheng, B. (2010). Model checking a model checker: A code contract combined approach. International Conference on Formal Engineering Methods,

Tennison, J. (2004). Validating xml with schematron. In *Beginning XSLT* (pp. 626-660). Springer.

Tomczak, A., van Berlo, L., Krijnen, T., Borrmann, A., Bolpagni, M., Research, I. C. f., Building, I. i., & Construction. (2022). A review of methods to specify information requirements in digital construction projects. World Building Congress WBC2022; International Council for Research and Innovation in Building and Construction, Ed,

W3C. (2012). W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. Retrieved: https://www.w3.org/TR/xmlschema11-1/. Accessed: 30.10.2022

W3C. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition). Retrieved: https://www.w3.org/TR/xml/. Accessed: 30.10.2022

Weise, M., Nisbet, N., Liebich, T., & Benghi, C. (2017). IFC model checking based on mvdXML 1.1. In *eWork and eBusiness in Architecture, Engineering and Construction* (pp. 19-26). CRC Press.

Windisch, R., Katranuschkov, P., & Scherer, R. (2012). A generic filter framework for consistent generation of BIM-based model views. Proceedings of the 2012 eg-ice Workshop,

Wu, C. T. (2006). *An Introduction to object-oriented programming with Java TM*. Mcgraw-Hill Incorporated.

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

Zhang, C., Beetz, J., & de Vries, B. (2013). Towards model view definition on semantic level: A state of the art review. Proceedings of the 20th International Workshop: Intelligent Computing in Engineering,

Zhang, C., Beetz, J., & Weise, M. (2014). Model view checking: automated validation for IFC building models. *eWork and eBusiness in Architecture, Engineering and Construction: ECPPM*, *14*.

Zhang, C., Beetz, J., & Weise, M. (2015). Interoperable validation for IFC building models using open standards. *Journal of Information Technology in Construction (ITcon)*, *20*(2), 24-39.

Appendix A

Furniture Example

A.1 Furniture XSD

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" tar-
getNamespace="furniture.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:fur="furniture.xsd">
  <xs:element name="furniture" type="fur:Furniture"/>

  <xs:complexType name="Furniture">
    <xs:sequence>
      <xs:element name="Chairs">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="fur:Chair" name="Chair" maxOccurs="unbounded"
minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Component">
    <!--Base class called component-->
    <xs:sequence>
      <xs:element name="Structure" type="fur:Material" maxOccurs="1" minOc-
curs="0"/>
      <xs:element name="Cushion" type="fur:Material" maxOccurs="1" minOc-
curs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="Color" use="optional"/>
    <xs:attribute type="xs:string" name="ID" use="required"/>
  </xs:complexType>

  <xs:simpleType name="Category">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Textile"/>
      <xs:enumeration value="Metal"/>
      <xs:enumeration value="Wood"/>
      <xs:enumeration value="Leather"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="Material">
    <xs:attribute type="fur:Category" name="Category" use="required"/>
    <xs:attribute type="xs:string" name="Name" use="required"/>
  </xs:complexType>

  <xs:complexType name="Chair">
    <xs:complexContent>
      <xs:extension base="fur:Component">
        <!--Chair inherits the features from base class component-->
```

```
        <xs:sequence>
          <xs:element name="Armrests">
            <xs:complexType>
              <xs:sequence>
                <xs:element type="fur:Armrest" name="Armrest" maxOccurs="2"
minOccurs="0"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Legs">
            <xs:complexType>
              <xs:sequence>
                <xs:element type="fur:Leg" name="Leg" maxOccurs="6" minOc-
curs="3"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="Production" type="fur:Production" maxOccurs="1"
minOccurs="0"/>
        </xs:sequence>
        <xs:attribute type="xs:boolean" name="HasBackrest" use="required"/>
        <xs:attribute type="xs:boolean" name="IsReclining" use="required"/>
        <xs:attribute type="xs:boolean" name="HasHeadrest" use="required"/>
        <xs:attribute type="xs:double" name="Length" use="optional"/>
        <xs:attribute type="xs:double" name="Width" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="Armrest">
    <!--Armrest inherits the features from base class component-->
    <xs:complexContent>
      <xs:extension base="fur:Component">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="Leg">
    <!--Leg inherits the features from base class component-->
    <xs:complexContent>
      <xs:extension base="fur:Component">
        <xs:attribute type="xs:boolean" name="HasWheel" use="required"/>
        <xs:attribute type="xs:boolean" name="IsSwiveling" use="required"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Production">
    <xs:attribute type="xs:boolean" name="IsAntique" use="optional"/>
    <xs:attribute type="xs:integer" name="ProductionYear" use="optional"/>
    <xs:attribute type="xs:string" name="Country" use="optional"/>
    <xs:attribute type="xs:string" name="Manufacturer" use="optional"/>
  </xs:complexType>
</xs:schema>
```

A.2 Furniture Dataset

```
<?xml version="1.0" encoding="utf-8"?>
<furniture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="furniture.xsd">
  <Chairs>
    <Chair Color="Black" ID="1" HasBackrest="true" IsReclining="true" HasHead-
rest="true" Length="52.5" Width="48.5">
      <Cushion Category="Leather" Name="Faux Leather" />
      <Armrests />
      <Legs>
        <Leg Color="Black" ID="1.1" HasWheel="true" IsSwiveling="true" />
        <Leg Color="Black" ID="1.2" HasWheel="true" IsSwiveling="true" />
        <Leg Color="Black" ID="1.3" HasWheel="true" IsSwiveling="true" />
      </Legs>
    </Chair>
    <Chair Color="White" ID="2" HasBackrest="false" IsReclining="false" HasHead-
rest="true">
      <Structure Category="Metal" Name="Steel" />
      <Cushion Category="Leather" Name="Faux Leather" />
      <Armrests>
        <Armrest ID="2.1">
          <Structure Category="Metal" Name="Aluminium" />
          <Cushion Category="Leather" Name="Faux Leather" />
        </Armrest>
        <Armrest ID="2.2">
          <Structure Category="Metal" Name="Steel" />
          <Cushion Category="Leather" Name="Faux Leather" />
        </Armrest>
      </Armrests>
      <Legs>
        <Leg Color="Black" ID="2.1" HasWheel="true" IsSwiveling="false">
          <Structure Category="Wood" Name="Timber" />
        </Leg>
        <Leg Color="Black" ID="2.2" HasWheel="true" IsSwiveling="false">
          <Structure Category="Wood" Name="Timber" />
        </Leg>
        <Leg Color="Black" ID="2.3" HasWheel="true" IsSwiveling="false">
          <Structure Category="Wood" Name="Timber" />
        </Leg>
        <Leg Color="Black" ID="2.4" HasWheel="true" IsSwiveling="false">
          <Structure Category="Wood" Name="Timber" />
        </Leg>
      </Legs>
      <Production IsAntique="true" ProductionYear="1960" Country="England" Manu-
facturer="Thomas Chippendale" />
    </Chair>
    <Chair Color="Black" ID="3" HasBackrest="true" IsReclining="true" HasHead-
rest="true" Length="50.5">
      <Structure Category="Metal" Name="Steel" />
      <Cushion Category="Textile" Name="Silk" />
      <Armrests>
        <Armrest ID="3.1">
          <Cushion Category="Leather" Name="Faux Leather" />
        </Armrest>
      </Armrests>
      <Legs>
        <Leg ID="3.1" HasWheel="false" IsSwiveling="false">
          <Structure Category="Metal" Name="Steel" />
        </Leg>
        <Leg ID="3.2" HasWheel="false" IsSwiveling="false">
          <Structure Category="Metal" Name="Steel" />
```

```
        </Leg>
        <Leg ID="3.3" HasWheel="true" IsSwiveling="false">
          <Structure Category="Metal" Name="Steel" />
        </Leg>
      </Legs>
      <Production IsAntique="true" ProductionYear="1940" Country="Germany" />
    </Chair>
    <Chair Color="Blue" ID="4" HasBackrest="true" IsReclining="true" HasHead-
rest="true" Length="55.5" Width="50.5">
      <Structure Category="Metal" Name="Steel" />
      <Cushion Category="Leather" Name="Faux Leather" />
      <Armrests>
        <Armrest ID="4.1">
          <Structure Category="Metal" Name="Steel" />
          <Cushion Category="Leather" Name="Faux Leather" />
        </Armrest>
      </Armrests>
      <Legs>
        <Leg Color="Blue" ID="4.1" HasWheel="true" IsSwiveling="true">
          <Structure Category="Metal" Name="Steel" />
        </Leg>
        <Leg Color="Blue" ID="4.2" HasWheel="true" IsSwiveling="true">
          <Structure Category="Metal" Name="Steel" />
        </Leg>
        <Leg Color="Blue" ID="4.3" HasWheel="true" IsSwiveling="true">
          <Structure Category="Metal" Name="Steel" />
        </Leg>
      </Legs>
      <Production IsAntique="true" ProductionYear="1940" Country="England" Manu-
facturer="Thomas Chippendale" />
    </Chair>
  </Chairs>
</furniture>
```

A.3 Furniture mvdXML File

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="9ba97c1d-c13c-4b4f-9581-
51565828e701" name="Furniture mvdXML" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References IdPrefix="Armrests_">
                <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
            <EntityRule EntityName="Leg">
              <References IdPrefix="Legs_">
                <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              </References>
```

```
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="5d2e52c2-cc53-441d-99b2-58df8312dda8" name="Arm-
restConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Arm-
rest">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
   </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
```

```
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="9f10223f-0da5-4536-9e17-0f96f13dd6e8" name="Compo-
nentConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Compo-
nent">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="83f5229e-f411-466e-b25b-b89d04666e1a" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="7a273105-4e0a-4349-a59e-
304669341d75" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7" name="Chair Has-
Backrest, then IsReclining, has Armrests, HasHeadrest" applicable-
RootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
```

```
            <TemplateRules>
              <TemplateRule Parameters="HasBackrest[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="fdc7fec5-6e23-4e99-b3e0-c84832662a6c" name="Chair Has-
Backrest, then IsReclining, has Armrests, HasHeadrest">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="IsReclining[Value]=TRUE AND Arm-
rests[Size]&gt;0 AND HasHeadrest[Value]=TRUE" Description="check if recline, arm-
rest, and headrest exists if chair has backrest" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="0c7b70bc-d061-4ff7-bb93-daf67c428ac1" name="Chair Pro-
duction" applicableRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="IsAntique[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="2300768b-1297-4bb8-b348-978c449633b7" name="Chair Pro-
duction">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Country[Exists]=TRUE AND Manufac-
turer[Exists]=TRUE AND ProductionYear[Exists]=TRUE" Description="check if produc-
tion year, country of orgin, and manufacturer attributes exists if the chair is
antique" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="86f2a7d8-8e90-485a-a700-f3a8d0f0693b" name="Chair
Structure=steel, then Cushion=faux leather" applicableRootEntity="Chair">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="StructureMaterialName[Value]='Steel'" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="cf38f321-63cc-4c15-93e3-09291441552d" name="Chair
Structure=steel, then Cushion=faux leather">
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
```

```
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="CushionMaterialName[Value]='Faux Leath-
er'" Description="check if chair strcuctal material is steel then chair cushioned
material must be leather" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="be790918-5b48-416c-9dd2-ccbe1b51f86a" name="Chair
length and width" applicableRootEntity="Chair">
          <Concepts>
            <Concept uuid="3aa515cf-c928-4f8a-8fb3-89b252c5e2b1" name="Chair
length and width">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Length[Value]&gt;=52.5 AND
Width[Value]&gt;=48.5" Description="check that the length is equal or greater
than 52.5 and width is equal or greater than 48.5" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="Component
Cushion exists, then Structure exists" applicableRootEntity="Component">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="Cushion[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="Component
Cushion exists, then Structure exists">
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Structure[Exists]=TRUE" Descrip-
tion="check that structural material exists when cushioned material exists" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="74884d5c-d231-4204-aaeb-3a3fbc13049d" name="Leg
HasWheel, then IsSwiveling" applicableRootEntity="Leg">
          <Applicability>
            <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
            <TemplateRules>
              <TemplateRule Parameters="HasWheel[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
```

```
          <Concepts>
            <Concept uuid="2c5eaeec-cc64-45da-9cc6-e2f7257f59d1" name="Leg
HasWheel, then IsSwiveling">
              <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="IsSwiveling[Value]=TRUE" Descrip-
tion="check that the wheels can swivel" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="811c5e38-684d-4684-a54d-a5c4cc0c7517" name="Armrest
StructureCategory is metal, then steel" applicableRootEntity="Armrest">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="StructureCategory[Value]='Metal'" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="9ba7e275-5012-43a0-a63c-714c8b5065ee" name="Armrest
StructureCtegory is metal, then steel">
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="StructureMaterialName[Value]='Steel'"
Description="check that StructureMaterialName is steel when StructureCategory is
metal" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="81e78ab9-5560-4ce3-a4d6-7cad75f8920b" name="Chair and
Armrests Cushion similar" applicableRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="Armrests[Size]&amp;gt;0" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="ac7f9692-913f-4992-b276-b3b9a3bddb0d" name="Chair and
Armrests Cushion similar">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
```

```
                <TemplateRule Parameters="Armrests_CushionMaterial-
Name[Value]=CushionMaterialName[Value]" Description="check if armrests cushion
material are the same as the cahir" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="6cc834bf-aced-4f7d-9e20-7c6fd09096da" name="Chair and
Leg colors similar" applicableRootEntity="Chair">
          <Concepts>
            <Concept uuid="99c1cc18-ef3b-4ef6-9049-58e8d4ac14ef" name="Chair and
Leg colors similar">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="7a273105-4e0a-4349-a59e-
304669341d75" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Color[Value]=Legs_Color[Value]" De-
scription="check that the color of chair and legs are same" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ComponentRuleIDUniqueness

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="b0be1b17-fd1b-4115-b116-
75bc202dac47" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="9f10223f-0da5-4536-9e17-0f96f13dd6e8" name="Compo-
nentConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Compo-
nent">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="MaterialName" />
                <AttributeRule AttributeName="Category" RuleID="Category" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="MaterialName" />
                <AttributeRule AttributeName="Category" RuleID="Category" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="7b7d205a-8217-40eb-b417-0a6de675f046" name="ComponentMod-
elView" applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applica-
bleRootEntity="Component">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="Cushion[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
```

```
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Structure[Exists]=TRUE" Descrip-
tion="check that structural material exists when cushioned material exists" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ComponentRuleIDExistence

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="eff7a740-9657-4106-a9d3-
95224ae2ccb4" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="9f10223f-0da5-4536-9e17-0f96f13dd6e8" name="Compo-
nentConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Compo-
nent">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name"  />
                <AttributeRule AttributeName="Category"  />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name"/>
                <AttributeRule AttributeName="Category" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="979934da-8b06-4d07-ae9a-756e5918a66a" name="ComponentMod-
elView" applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applica-
bleRootEntity="Component">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="Cushion[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
```

```
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <Requirements>
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="Structure[Exists]=TRUE" Descrip-
tion="check that structural material exists when cushioned material exists" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
    </Roots>
  </ModelView>
 </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairIDPrefixUniqueness

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="9fa9933f-b96f-4ca2-9d0f-
17c139af9d31" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References IdPrefix="Armrests_">
                <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
```

```
            <EntityRule EntityName="Leg">
              <References IdPrefix="Armrests_">
                <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="5d2e52c2-cc53-441d-99b2-58df8312dda8" name="Arm-
restConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Arm-
rest">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
```

```
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="3e86a57c-91df-4c8c-8d4e-603d7c1bca9f" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="cc841c98-136d-4791-a8cb-ee8204e0bc4b" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="Armrests[Size]&gt;0" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="6d41b0ee-e4e2-46cb-8376-1f025bba0b8f" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Armrests.CushionMaterial-
Name[Value]=CushionMaterialName[Value]" Description="check if armrests cushion
material are the same as the cahir" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairIDPrefixExistence

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="16b39f36-0939-47fd-9437-
09146be1bb4d" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References>
                <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
```

```
            <EntityRule EntityName="Leg">
              <References>
                <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="5d2e52c2-cc53-441d-99b2-58df8312dda8" name="Arm-
restConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Arm-
rest">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
```

```
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="6430db01-7b42-43d6-9771-3c044fc8ec09" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="cc841c98-136d-4791-a8cb-ee8204e0bc4b" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="Armrests[Size]&gt;0" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="6d41b0ee-e4e2-46cb-8376-1f025bba0b8f" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Armrests.CushionMaterial-
Name[Value]=CushionMaterialName[Value]" Description="check if armrests cushion
material are the same as the cahir" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ComponentMetricValue

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="a6f819ae-8715-465d-b6a7-
d4b76298efbc" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="9f10223f-0da5-4536-9e17-0f96f13dd6e8" name="Compo-
nentConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Compo-
nent">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="7feec79c-0c8f-45db-9db9-bcad6fcf6532" name="ComponentMod-
elView" applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applica-
bleRootEntity="Component">
          <Applicability>
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <TemplateRules>
              <TemplateRule Parameters="Cushion[Valu]=TRUE" />
```

```
            <TemplateRule Parameters="CushionCategory[]='Leather'" />
          </TemplateRules>
        </Applicability>
        <Concepts>
          <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
            <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
            <Requirements>
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules operator="or">
              <TemplateRule Parameters="Structure[Exists]=TRUE AND Structure-
Category[yes]=TRUE"/>
              <TemplateRules operator="and">
                <TemplateRule Parameters="Structure[Exists]=TRUE AND Structure-
Category[]='Metal'" />
                <TemplateRule Parameters="StructureMaterialName[Exist]=TRUE"/>
              </TemplateRules>
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
    </Roots>
  </ModelView>
 </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ComponentParametersAttributes

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="a6f819ae-8715-465d-b6a7-
d4b76298efbc" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="9f10223f-0da5-4536-9e17-0f96f13dd6e8" name="Compo-
nentConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Compo-
nent">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="7feec79c-0c8f-45db-9db9-bcad6fcf6532" name="ComponentMod-
elView" applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="31adb6df-f558-4594-b9f3-ed7e3e352bd9" name="" applica-
bleRootEntity="Component">
          <Concepts>
            <Concept uuid="afe6bbcc-91a8-477f-95ff-318a2c832e0c" name="">
              <Template ref="9f10223f-0da5-4536-9e17-0f96f13dd6e8" />
              <Requirements>
```

```
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules operator="or">
                  <TemplateRule Parameters="Category[Value]='Metal'" />
                  <TemplateRule Parameters="StructureName[Exists]=TRUE"/>
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairTemplateReference

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="df5a3296-7d6f-42c7-97ea-
80cbcc8fc3e7" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References IdPrefix="Armrests_">
                <Template ref=""/>
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
```

```
            <EntityRule EntityName="Leg">
              <References IdPrefix="Legs_">
                <Template ref=""/>
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="5d2e52c2-cc53-441d-99b2-58df8312dda8" name="Arm-
restConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Arm-
rest">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
```

```
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="11797c75-bc2d-4390-8419-d633bc206bb4" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="cc841c98-136d-4791-a8cb-ee8204e0bc4b" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="" />
            <TemplateRules>
              <TemplateRule Parameters="Armrests[Size]&gt;0" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="6d41b0ee-e4e2-46cb-8376-1f025bba0b8f" name="">
              <Template ref="" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Armrests.CushionMaterial-
Name[Value]=CushionMaterialName[Value]" Description="check if armrests cushion
material are the same as the cahir" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

Chair-LegEntityReference

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="f65063a2-b4b2-4da5-8cb8-
bb26d33bd38b" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
            <EntityRule EntityName="Leg">
              <References IdPrefix="Legs_">
                <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
```

```
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="6a9f3a29-ac49-4f00-9830-04b6a0353e8b" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="0c7b70bc-d061-4ff7-bb93-daf67c428ac1" name="" applica-
bleRootEntity="Leg">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="IsAntique[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="2300768b-1297-4bb8-b348-978c449633b7" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
```

```
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Country[Exists]=TRUE AND Manufac-
turer[Exists]=TRUE AND ProductionYear[Exists]=TRUE" Description="check if produc-
tion year, country of orgin, and manufacturer attributes exists if the chair is
antique" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairReferencedConceptTemplates

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="7f06d0ef-1246-41b5-bb10-
0b0c12e511d2" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
         <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References IdPrefix="Armrests_">
                <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
```

```
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="8106005c-8964-476a-9c53-d652207da7b2" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="0c7b70bc-d061-4ff7-bb93-daf67c428ac1" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d95" />
            <TemplateRules>
              <TemplateRule Parameters="IsAntique[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="2300768b-1297-4bb8-b348-978c449633b7" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d95" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Country[Exists]=TRUE AND Manufac-
turer[Exists]=TRUE AND ProductionYear[Exists]=TRUE" Description="check if produc-
tion year, country of orgin, and manufacturer attributes exists if the chair is
antique" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairConstraints

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="266ad15b-9dac-46de-b500-
bdda6e0e6d79" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Armrests" RuleID="Armrests">
          <EntityRules>
            <EntityRule EntityName="Armrest">
              <References IdPrefix="Armrests_">
                <Template ref="5d2e52c2-cc53-441d-99b2-58df8312dda8" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Legs" RuleID="Legs">
          <EntityRules>
```

```
            <EntityRule EntityName="Leg">
              <References IdPrefix="Legs_">
                <Template ref="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="5d2e52c2-cc53-441d-99b2-58df8312dda8" name="Arm-
restConceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Arm-
rest">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="28edd9a1-fdd2-4298-9a94-49c7ccc031b3" name="LegCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Leg">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
```

```
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="HasWheel" RuleID="HasWheel" />
        <AttributeRule AttributeName="IsSwiveling" RuleID="IsSwiveling" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="c342feeb-dd6f-48a2-bccd-b31d4674d342" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="fdc7fec5-6e23-4e99-b3e0-c84832662a6c" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="IsReclining[Value]=TRUE AND Arm-
rests[Size]&gt;0 AND HasHeadrest[Value]=TRUE" Description="check if recline, arm-
rest, and headrest exists if chair has backrest" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="0c7b70bc-d061-4ff7-bb93-daf67c428ac1" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="IsAntique[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="12ad3dad-264a-4f6e-ac4d-df669d03b9a2" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
```

```
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Manufacturer[Exists]=TRUE"  Descrip-
tion="Manufacture exists AND"/>
                <TemplateRules operator="or">
                  <TemplateRule Parameters="" Description="IsAntique OR Country
exists" />
                  <TemplateRule Parameters="Country[Exists]=TRUE"  Descrip-
tion="IsAntique OR Country exists"/>
                </TemplateRules>
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairName

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="c8a623f4-7d9e-428a-9c56-
74e0cdce9923" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="" applica-
bleSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
```

```
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
  </ConceptTemplate>
 </Templates>
 <Views>
  <ModelView uuid="6fe45497-9985-495a-a1a9-fa8c64f457b9" name="" applica-
bleSchema="FurnitureSchema">
    <ExchangeRequirements>
      <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
    </ExchangeRequirements>
    <Roots>
      <ConceptRoot uuid="32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7" name="" applica-
bleRootEntity="Chair">
        <Applicability>
          <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
          <TemplateRules>
            <TemplateRule Parameters="HasBackrest[Value]=TRUE" />
          </TemplateRules>
        </Applicability>
        <Concepts>
          <Concept uuid="fdc7fec5-6e23-4e99-b3e0-c84832662a6c" name="">
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <Requirements>
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="IsReclining[Value]=TRUE AND Arm-
rests[Size]&gt;0 AND HasHeadrest[Value]=TRUE" Description="check if recline, arm-
rest, and headrest exists if chair has backrest" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
    </Roots>
  </ModelView>
 </Views>
</mvdXML>
```

A.4 Furniture mvdXML Incorrect Files

ChairOperatorExistence

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="0b0a12fb-1951-44dc-8eba-
4769c0758d13" name="" xmlns="http://buildingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" name="ChairCon-
ceptTemplate" applicableSchema="FurnitureSchema" applicableEntity="Chair">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="ID" RuleID="ID" />
        <AttributeRule AttributeName="HasBackrest" RuleID="HasBackrest" />
        <AttributeRule AttributeName="IsReclining" RuleID="IsReclining" />
        <AttributeRule AttributeName="HasHeadrest" RuleID="HasHeadrest" />
        <AttributeRule AttributeName="Color" RuleID="Color" />
        <AttributeRule AttributeName="Length" RuleID="Length" />
        <AttributeRule AttributeName="Width" RuleID="Width" />
        <AttributeRule AttributeName="Structure" RuleID="Structure">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="StructureMaterial-
Name" />
                <AttributeRule AttributeName="Category" RuleID="StructureCate-
gory" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Cushion" RuleID="Cushion">
          <EntityRules>
            <EntityRule EntityName="Material">
              <AttributeRules>
                <AttributeRule AttributeName="Name" RuleID="CushionMaterialName"
/>
                <AttributeRule AttributeName="Category" RuleID="CushionCategory"
/>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
        <AttributeRule AttributeName="Production" RuleID="Production">
          <EntityRules>
            <EntityRule EntityName="Production">
              <AttributeRules>
                <AttributeRule AttributeName="IsAntique" RuleID="IsAntique" />
                <AttributeRule AttributeName="ProductionYear" RuleID="Produc-
tionYear" />
                <AttributeRule AttributeName="Country" RuleID="Country" />
                <AttributeRule AttributeName="Manufacturer" RuleID="Manufacturer"
/>
              </AttributeRules>
```

```
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="1ce7b63b-9339-4b39-93b6-28760ef08eb2" name="ChairModelView"
applicableSchema="FurnitureSchema">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" name="Chair Checks" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="0c7b70bc-d061-4ff7-bb93-daf67c428ac1" name="" applica-
bleRootEntity="Chair">
          <Applicability>
            <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
            <TemplateRules>
              <TemplateRule Parameters="IsAntique[Value]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="12ad3dad-264a-4f6e-ac4d-df669d03b9a2" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Manufacturer[Exists]=TRUE"  Descrip-
tion="Manufacture exists AND"/>
                <TemplateRules>
                  <TemplateRule Parameters="ProductionYear[Exists]=TRUE" Descrip-
tion="IsAntique OR Country exists" />
                  <TemplateRule Parameters="Country[Exists]=TRUE"  Descrip-
tion="IsAntique OR Country exists"/>
                </TemplateRules>
              </TemplateRules>
            </Concept>
            <Concept uuid="2300768b-1297-4bb8-b348-978c449633b7" name="">
              <Template ref="ab8ea1a0-3610-4e2f-b3d6-0f6414522d94" />
              <Requirements>
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="d6e6c274-9087-48d7-b2aa-
7bf062a4dc46" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Country[Exists]=TRUE AND Manufac-
turer[Exists]=TRUE AND ProductionYear[Exists]=TRUE" Description="check if produc-
tion year, country of orgin, and manufacturer attributes exists if the chair is
antique" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

Appendix B

MvdXML Checks

B.1 RuleID

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="f5a92a6c-da0f-4333-92e7-
a67f988c9a33" name="RuleID check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="dd83efa6-88c4-4954-b378-7f6ffaaec311" name="RuleID
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="AttributeRule">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="RuleID" RuleID="RuleID" />
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="dd83efa6-88c4-4954-b378-7f6ffaaec312" name="RuleID
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="ConceptTem-
plate">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Rules">
          <EntityRules>
            <EntityRule EntityName="AttributeRule">
              <References>
                <Template ref="dd83efa6-88c4-4954-b378-7f6ffaaec311" />
              </References>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="446e5681-5773-4a35-a1ca-9d86873072da" name="RuleIDModelView"
applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="5b4e94ff-fc86-4df8-a651-
028f731c9dc2" name="RuleID check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="5d33acef-c6be-4b54-b9b8-a24ec0c4d043" name="RuleID ex-
istence" applicableRootEntity="AttributeRule">
          <Concepts>
            <Concept uuid="68ecc19c-cd8c-4a38-b0d2-8bf16d2b6455" name="RuleID ex-
istence">
```

```
              <Template ref="dd83efa6-88c4-4954-b378-7f6ffaaec311" />
              <Requirements>
                <Requirement exchangeRequirement="5b4e94ff-fc86-4df8-a651-
028f731c9dc2" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="5b4e94ff-fc86-4df8-a651-
028f731c9dc2" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="RuleID[Exists]=TRUE" Description="check
if RuleID exists for each AttributRule" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="112c33e2-366c-4ab8-9491-47f881bbc07e" name="RuleID
uniqueness" applicableRootEntity="ConceptTemplate">
          <Concepts>
            <Concept uuid="e5eaa92e-d530-414f-bf86-c50a3b84fbe2" name="RuleID
uniqueness">
              <Template ref="dd83efa6-88c4-4954-b378-7f6ffaaec312" />
              <Requirements>
                <Requirement exchangeRequirement="5b4e94ff-fc86-4df8-a651-
028f731c9dc2" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="5b4e94ff-fc86-4df8-a651-
028f731c9dc2" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="RuleID[Unique]=TRUE" Description="check
if RuleID is unique" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.2 IdPrefix

```
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="b4289196-e8be-4a79-a27e-
8aa14e9c1333" name="IdPrefix check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="4c31b9d9-3a83-4120-a56d-9b116593e02e" name="IdPrefix
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="EntityRule">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="References" RuleID="References">
          <EntityRules>
            <EntityRule EntityName="EntityRuleReferences">
              <AttributeRules>
                <AttributeRule AttributeName="IdPrefix" RuleID="IdPrefix" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="4c31b9d9-3a83-4120-a56d-9b116593e02f" name="IdPrefix
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="ConceptTem-
plate">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Rules">
          <EntityRules>
            <EntityRule EntityName="AttributeRule">
              <AttributeRules>
                <AttributeRule AttributeName="EntityRules">
                  <EntityRules>
                    <EntityRule EntityName="AttributeRuleEntityRules">
                      <AttributeRules>
                        <AttributeRule AttributeName="EntityRule">
                          <EntityRules>
                            <EntityRule EntityName="EntityRule">
                              <References>
                                <Template ref="4c31b9d9-3a83-4120-a56d-
9b116593e02e" />
                              </References>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
                      </AttributeRules>
                    </EntityRule>
                  </EntityRules>
                </AttributeRule>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
```

```
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="af93039d-0e9e-4c36-a467-5d94ec2b2546" name="IdPrefix Mod-
elView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="4b52ac1c-f3fb-41f1-b804-
467f1ab32197" name="IdPrefix check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="0a48b28a-c7b4-4e53-8153-d6042aaf0ebc" name="IdPrefix
existence" applicableRootEntity="EntityRule">
          <Applicability>
            <Template ref="4c31b9d9-3a83-4120-a56d-9b116593e02e" />
            <TemplateRules>
              <TemplateRule Parameters="References[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="cb576fee-b25f-4d51-aa07-cd436851317d" name="IdPrefix
existence">
              <Template ref="4c31b9d9-3a83-4120-a56d-9b116593e02e" />
              <Requirements>
                <Requirement exchangeRequirement="4b52ac1c-f3fb-41f1-b804-
467f1ab32197" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4b52ac1c-f3fb-41f1-b804-
467f1ab32197" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="IdPrefix[Exists]=TRUE" Descrip-
tion="check the existence of IdPrefix when References exist" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="0a48b28a-c7b4-4e53-8153-d6042aaf0ebd" name="IdPrefix
uniqueness" applicableRootEntity="ConceptTemplate">
          <Applicability>
            <Template ref="4c31b9d9-3a83-4120-a56d-9b116593e02f" />
            <TemplateRules>
              <TemplateRule Parameters="IdPrefix[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="cb576fee-b25f-4d51-aa07-cd436851317c" name="IdPrefix
uniqueness">
              <Template ref="4c31b9d9-3a83-4120-a56d-9b116593e02f" />
              <Requirements>
                <Requirement exchangeRequirement="4b52ac1c-f3fb-41f1-b804-
467f1ab32197" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4b52ac1c-f3fb-41f1-b804-
467f1ab32197" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="IdPrefix[Unique]=TRUE" Descrip-
tion="check the uniqueness of IdPrefix when References exist" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.3 MetricValue

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="efa19fc4-d99b-4762-873a-
02f614f177b3" name="MetricValue check mvdXML" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="8e900bab-2693-4a44-88ac-5ba07a654ae7" name="Met-
ricValue ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Tem-
plateRulesTemplateRule">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Parameters" RuleID="Parameters" />
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="37d669a7-9e7d-4823-bf75-0ede48ec05ea" name="MetricValue Mod-
elView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="c5e9d969-adae-4c05-b6d1-
646ee8cac94e" name="MetricValue check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="12e23e60-73af-48c3-ba5b-889fc9ea88b0" name="Met-
ricValue" applicableRootEntity="TemplateRulesTemplateRule">
          <Concepts>
            <Concept uuid="3bb3ea16-af65-4806-9f42-4c53464bfecc" name="Met-
ricValue">
              <Template ref="8e900bab-2693-4a44-88ac-5ba07a654ae7" />
              <Requirements>
                <Requirement exchangeRequirement="c5e9d969-adae-4c05-b6d1-
646ee8cac94e" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="c5e9d969-adae-4c05-b6d1-
646ee8cac94e" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Parameters[Value]!=reg'.*(\\[\\]).*'
AND Parame-
ters[Value]=reg'.*(\\[Value\\]|\\[VALUE\\]|\\[Size\\]|\\[SIZE\\]|\\[Type\\]|\\[TY
PE\\]|\\[Unique\\]|\\[UNIQUE\\]|\\[Exists\\]|\\[EXISTS\\]).*'" Description="check
the existence of a metric value for each attribute in the parameters string" />
                <TemplateRule Parameters="Parameters[Value]!=reg'.*(\\[[a-
z]+\\]).*' AND Parame-
ters[Value]=reg'.*(\\[Value\\]|\\[VALUE\\]|\\[Size\\]|\\[SIZE\\]|\\[Type\\]|\\[TY
PE\\]|\\[Unique\\]|\\[UNIQUE\\]|\\[Exists\\]|\\[EXISTS\\]).*'" Description="check
the existence of a metric value for each attribute in the parameters string" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.4 ParametersAttributes

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="f6510261-fc19-4f59-b352-
7fe704f89eef" name="Parameters Attributes" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ca46edea-a1e7-4ed1-84d0-8a8a5366759a" name="Parameters
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="mvdXML">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Views">
          <EntityRules>
            <EntityRule EntityName="ModelView">
              <AttributeRules>
                <AttributeRule AttributeName="Roots">
                  <EntityRules>
                    <EntityRule EntityName="ConceptRoot">
                      <AttributeRules>
                        <AttributeRule AttributeName="Applicability">
                          <EntityRules>
                            <EntityRule EntityName="ConceptRootApplicability">
                              <AttributeRules>
                                <AttributeRule AttributeName="TemplateRules">
                                  <EntityRules>
                                    <EntityRule EntityName="TemplateRules">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="Items">
                                          <EntityRules>
                                            <EntityRule EntityName="Templat-
eRulesTemplateRule">
                                              <AttributeRules>
                                                <AttributeRule AttributeName="Pa-
rameters" RuleID="ApplicabilityParameters" />
                                              </AttributeRules>
                                            </EntityRule>
                                          </EntityRules>
                                        </AttributeRule>
                                      </AttributeRules>
                                    </EntityRule>
                                  </EntityRules>
                                </AttributeRule>
                              </AttributeRules>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
                        <AttributeRule AttributeName="Concepts">
                          <EntityRules>
                            <EntityRule EntityName="Concept">
                              <AttributeRules>
                                <AttributeRule AttributeName="TemplateRules">
                                  <EntityRules>
                                    <EntityRule EntityName="TemplateRules">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="Items">
                                          <EntityRules>
```

```
                                                <EntityRule EntityName="Templat-
eRulesTemplateRule">
                                                  <AttributeRules>
                                                    <AttributeRule AttributeName="Pa-
rameters" RuleID="ConceptsParameters" />
                                                  </AttributeRules>
                                                </EntityRule>
                                              </EntityRules>
                                            </AttributeRule>
                                          </AttributeRules>
                                        </EntityRule>
                                      </EntityRules>
                                    </AttributeRule>
                                  </AttributeRules>
                                </EntityRule>
                              </EntityRules>
                            </AttributeRule>
                          </AttributeRules>
                        </EntityRule>
                      </EntityRules>
                    </AttributeRule>
                  </AttributeRules>
                </EntityRule>
              </EntityRules>
            </AttributeRule>
            <AttributeRule AttributeName="Templates">
              <EntityRules>
                <EntityRule EntityName="ConceptTemplate">
                  <AttributeRules>
                    <AttributeRule AttributeName="Rules">
                      <EntityRules>
                        <EntityRule EntityName="AttributeRule">
                          <AttributeRules>
                            <AttributeRule AttributeName="RuleID" RuleID="RuleID" />
                          </AttributeRules>
                        </EntityRule>
                      </EntityRules>
                    </AttributeRule>
                  </AttributeRules>
                </EntityRule>
              </EntityRules>
            </AttributeRule>
          </Rules>
        </ConceptTemplate>
      </Templates>
      <Views>
        <ModelView uuid="28f35c0d-b708-4737-8ed1-5c4d705bf284" name="ReferencedAt-
tributesModelView" applicableSchema="mvdXML_V1.1">
          <ExchangeRequirements>
            <ExchangeRequirement applicability="both" uuid="cf892156-6c33-4c1d-a960-
060fe2b67474" name="ReferencedAttributes check" />
          </ExchangeRequirements>
          <Roots>
            <ConceptRoot uuid="81a939d4-7568-460e-b212-346efb5d0af4" name="Applica-
bility parameters attributes" applicableRootEntity="mvdXML">
              <Concepts>
                <Concept uuid="714da4f2-4aca-498b-9094-6141b5f44747" name="Applica-
bility parameters attributes">
                  <Template ref="ca46edea-a1e7-4ed1-84d0-8a8a5366759a" />
                  <Requirements>
                    <Requirement exchangeRequirement="cf892156-6c33-4c1d-a960-
060fe2b67474" requirement="mandatory" applicability="import" />
                    <Requirement exchangeRequirement="cf892156-6c33-4c1d-a960-
060fe2b67474" requirement="mandatory" applicability="export" />
```

```
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="ApplicabilityParame-
ters[Value]=reg'.*(RuleID[Value]).*'" Description="check the existence of the at-
tributes in the concept template" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
      <ConceptRoot uuid="f1d6e1cd-4007-44ae-aaa9-ab3960555501" name="Concept
parameters attributes" applicableRootEntity="mvdXML">
        <Concepts>
          <Concept uuid="d6abc66a-b390-456b-bb40-43e679fdf2d1" name="Concept
parameters attributes">
            <Template ref="ca46edea-a1e7-4ed1-84d0-8a8a5366759a" />
            <Requirements>
              <Requirement exchangeRequirement="cf892156-6c33-4c1d-a960-
060fe2b67474" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="cf892156-6c33-4c1d-a960-
060fe2b67474" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="ConceptsParame-
ters[Value]=reg'.*(RuleID[Value]).*'" Description="check the existence of the at-
tributes in the concept template" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
    </Roots>
  </ModelView>
 </Views>
</mvdXML>
```

B.5 TemplateReference

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="af1b116d-bc06-46d1-a955-
8532c1cd1652" name="@ref Check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="ad9aeda2-ad85-431b-a95b-7ea56fb52b96" name="Tem-
plateReference ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="ConceptRoot">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Applicability" RuleID="Applicability">
          <EntityRules>
            <EntityRule EntityName="ConceptRootApplicability">
              <AttributeRules>
                <AttributeRule AttributeName="Template" RuleID="ApplicabilityTem-
plate">
                  <EntityRules>
                    <EntityRule EntityName="GenericReference">
                      <AttributeRules>
                        <AttributeRule AttributeName="ref" RuleID="Applicabil-
ityref" />
                      </AttributeRules>
                    </EntityRule>
                  </EntityRules>
                </AttributeRule>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="ad9aeda2-ad85-431b-a95b-7ea56fb52b97" name="Tem-
plateReference ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="Concept">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Template">
          <EntityRules>
            <EntityRule EntityName="GenericReference">
              <AttributeRules>
                <AttributeRule AttributeName="ref" RuleID="Conceptref" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="c9d40546-597e-41a5-9f1f-2f2ef50ca54c" name="EntityRule
reference ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Enti-
tyRule">
      <Definitions>
```

```
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="References" RuleID="References">
          <EntityRules>
            <EntityRule EntityName="EntityRuleReferences">
              <AttributeRules>
                <AttributeRule AttributeName="Template">
                  <EntityRules>
                    <EntityRule EntityName="GenericReference">
                      <AttributeRules>
                        <AttributeRule AttributeName="ref" RuleID="ref" />
                      </AttributeRules>
                    </EntityRule>
                  </EntityRules>
                </AttributeRule>
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="5a621057-3734-41e2-ba04-b0fb47a71f6b" name="TemplateRefer-
ence ModelView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" name="Template reference check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="2e84ff23-2825-49e8-a92e-8af75bb92753" name="Applica-
bility template reference" applicableRootEntity="ConceptRoot">
          <Applicability>
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b96" />
            <TemplateRules>
              <TemplateRule Parameters="Applicability[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="b45128ae-dc95-4de3-ad2d-6e2d8d2a9042" name="Applica-
bility template reference">
              <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b96" />
              <Requirements>
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Applicabilityref[Value]!='' Descrip-
tion="check the existence Template ref when Applicability exist" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="d80f9f4a-19f5-4585-aaed-1fed551ad8f7" name="Concept
template reference" applicableRootEntity="Concept">
          <Concepts>
            <Concept uuid="a71e6bfd-6ff5-4000-8966-7c737830fab7" name="Concept
template reference">
              <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b97" />
```

```
              <Requirements>
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="Conceptref[Value]!='' Descrip-
tion="check the existence of Template ref" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="44401324-8c79-4cfc-b40d-29a105f4de61" name="Entity
template reference" applicableRootEntity="EntityRule">
          <Applicability>
            <Template ref="c9d40546-597e-41a5-9f1f-2f2ef50ca54c" />
            <TemplateRules>
              <TemplateRule Parameters="References[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="4caf23ec-c495-497e-94c1-b35995686d61" name="Entity
template reference">
              <Template ref="c9d40546-597e-41a5-9f1f-2f2ef50ca54c" />
              <Requirements>
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="a00f80e9-1e8f-417d-b10e-
d8865ef6252e" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="ref[Value]!='' Description="check that
the ref value exists if references exist" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.6 EntityReference

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="64c2925d-0a75-401e-9b7a-
460b755b2951" name="EntityReference check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="c9d40546-597e-41a5-9f1f-2f2ef50ca54b" name="Con-
ceptTemplate ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="mvdXML">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Templates">
          <EntityRules>
            <EntityRule EntityName="ConceptTemplate">
              <AttributeRules>
                <AttributeRule AttributeName="applicableEntity" RuleID="applica-
bleEntity" />
                <AttributeRule AttributeName="uuid" RuleID="uuid" />
                <AttributeRule AttributeName="Rules">
                  <EntityRules>
                    <EntityRule EntityName="AttributeRule">
                      <AttributeRules>
                        <AttributeRule AttributeName="EntityRules">
                          <EntityRules>
                            <EntityRule EntityName="AttributeRuleEntityRules">
                              <AttributeRules>
                                <AttributeRule AttributeName="EntityRule">
                                  <EntityRules>
                                    <EntityRule EntityName="EntityRule">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="References"
RuleID="References">
                                          <EntityRules>
                                            <EntityRule EntityName="EntityRul-
eReferences">
                                              <AttributeRules>
                                                <AttributeRule Attribu-
teName="Template">
                                                  <EntityRules>
                                                    <EntityRule EntityName="Ge-
nericReference">
                                                      <AttributeRules>
                                                        <AttributeRule Attribu-
teName="ref" RuleID="ref" />
                                                      </AttributeRules>
                                                    </EntityRule>
                                                  </EntityRules>
                                                </AttributeRule>
                                              </AttributeRules>
                                            </EntityRule>
                                          </EntityRules>
                                        </AttributeRule>
                                      </AttributeRules>
                                    </EntityRule>
                                  </EntityRules>
                                </AttributeRule>
                              </AttributeRules>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
```

```
                                </AttributeRules>
                              </EntityRule>
                            </EntityRules>
                          </AttributeRule>
                        </AttributeRules>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                </AttributeRules>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
        </Rules>
      </ConceptTemplate>
    </Templates>
    <Views>
      <ModelView uuid="0ffa2776-5e08-4cd1-a39b-d9000ddd0862" name="ConceptTem-
platesExistenceModelView" applicableSchema="mvdXML_V1.1">
        <ExchangeRequirements>
          <ExchangeRequirement applicability="both" uuid="d573aa26-526e-492a-ae5b-
65247e07e114" name="Existence of ConceptTemplates check" />
        </ExchangeRequirements>
        <Roots>
          <ConceptRoot uuid="44401324-8c79-4cfc-b40d-29a105f4de69" name="Entity
Reference" applicableRootEntity="mvdXML">
            <Concepts>
              <Concept uuid="4caf23ec-c495-497e-94c1-b35995686d69" name="check that
the ref value is referring to the correct concept template by checking the enti-
ties">
                <Template ref="c9d40546-597e-41a5-9f1f-2f2ef50ca54b" />
                <Requirements>
                  <Requirement exchangeRequirement="d573aa26-526e-492a-ae5b-
65247e07e114" requirement="mandatory" applicability="import" />
                  <Requirement exchangeRequirement="d573aa26-526e-492a-ae5b-
65247e07e114" requirement="mandatory" applicability="export" />
                </Requirements>
                <TemplateRules>
                  <TemplateRule Parameters="ref[Value]=uuid[Value] AND Enti-
tyName[Value]=applicableEntity[Value]" Description="check that the ref value is
referring to the correct concept template by checking the entities" />
                </TemplateRules>
              </Concept>
            </Concepts>
          </ConceptRoot>
        </Roots>
      </ModelView>
    </Views>
</mvdXML>
```

B.7 ReferencedConceptTemplates

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="51512520-03d6-4584-991d-
009fd8f105c0" name="ReferencedConceptTemplates check" xmlns="http://build-
ingsmart-tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="58c7bea0-2ff8-41f3-97a4-be9854616a94" name="Refer-
encedConceptTemplates ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="mvdXML">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Views">
          <EntityRules>
            <EntityRule EntityName="ModelView">
              <AttributeRules>
                <AttributeRule AttributeName="Roots">
                  <EntityRules>
                    <EntityRule EntityName="ConceptRoot">
                      <AttributeRules>
                        <AttributeRule AttributeName="Applicability" RuleID="Ap-
plicability">
                          <EntityRules>
                            <EntityRule EntityName="ConceptRootApplicability">
                              <AttributeRules>
                                <AttributeRule AttributeName="Template">
                                  <EntityRules>
                                    <EntityRule EntityName="GenericReference">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="ref"
RuleID="Applicabilityref" />
                                      </AttributeRules>
                                    </EntityRule>
                                  </EntityRules>
                                </AttributeRule>
                              </AttributeRules>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
                        <AttributeRule AttributeName="Concepts">
                          <EntityRules>
                            <EntityRule EntityName="Concept">
                              <AttributeRules>
                                <AttributeRule AttributeName="Template">
                                  <EntityRules>
                                    <EntityRule EntityName="GenericReference">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="ref"
RuleID="Conceptref" />
                                      </AttributeRules>
                                    </EntityRule>
                                  </EntityRules>
                                </AttributeRule>
                              </AttributeRules>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
```

```
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
      <AttributeRule AttributeName="Templates">
        <EntityRules>
          <EntityRule EntityName="ConceptTemplate">
            <AttributeRules>
              <AttributeRule AttributeName="uuid" RuleID="uuid" />
              <AttributeRule AttributeName="SubTemplates">
                <EntityRules>
                  <EntityRule EntityName="ConceptTemplate">
                    <AttributeRules>
                      <AttributeRule AttributeName="uuid" RuleID="suuid" />
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
              <AttributeRule AttributeName="Rules">
                <EntityRules>
                  <EntityRule EntityName="AttributeRule">
                    <AttributeRules>
                      <AttributeRule AttributeName="EntityRules">
                        <EntityRules>
                          <EntityRule EntityName="AttributeRuleEntityRules">
                            <AttributeRules>
                              <AttributeRule AttributeName="EntityRule">
                                <EntityRules>
                                  <EntityRule EntityName="EntityRule">
                                    <AttributeRules>
                                      <AttributeRule AttributeName="References"
RuleID="References">
                                        <EntityRules>
                                          <EntityRule EntityName="EntityRul-
eReferences">
                                            <AttributeRules>
                                              <AttributeRule Attribu-
teName="Template">
                                                <EntityRules>
                                                  <EntityRule EntityName="Ge-
nericReference">
                                                    <AttributeRules>
                                                      <AttributeRule Attribu-
teName="ref" RuleID="Entityref" />
                                                    </AttributeRules>
                                                  </EntityRule>
                                                </EntityRules>
                                              </AttributeRule>
                                            </AttributeRules>
                                          </EntityRule>
                                        </EntityRules>
                                      </AttributeRule>
                                    </AttributeRules>
                                  </EntityRule>
                                </EntityRules>
                              </AttributeRule>
                            </AttributeRules>
                          </EntityRule>
                        </EntityRules>
                      </AttributeRule>
```

```
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
  </ConceptTemplate>
</Templates>
<Views>
  <ModelView uuid="d1619f42-1410-48dd-aed4-dcbb21c9f6b6" name="ConceptTem-
platesExistence ModelView" applicableSchema="mvdXML_V1.1">
    <ExchangeRequirements>
      <ExchangeRequirement applicability="both" uuid="1496d96f-f4e2-400a-8f75-
c044c7dd6878" name="Existence of ConceptTemplates check" />
    </ExchangeRequirements>
    <Roots>
      <ConceptRoot uuid="b3d73a9a-3693-462a-bed6-20658ed48122" name="Existence
of referred ConceptTemplates check of Applicability in the mvdXML file" applica-
bleRootEntity="mvdXML">
        <Applicability>
          <Template ref="58c7bea0-2ff8-41f3-97a4-be9854616a94" />
          <TemplateRules>
            <TemplateRule Parameters="Applicability[Exists]=TRUE" />
          </TemplateRules>
        </Applicability>
        <Concepts>
          <Concept uuid="1b6168cd-1336-4c19-b9dc-4afc19ec77f9" name="check that
all referenced ConceptTemplates in applicability exists in the mvdxml file">
            <Template ref="58c7bea0-2ff8-41f3-97a4-be9854616a94" />
            <Requirements>
              <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules operator="or">
              <TemplateRule Parameters="Applicabilityref[Value]=uuid[Value]"
Description="check that all referenced ConceptTemplates in applicability exists
in the mvdxml file" />
              <TemplateRule Parameters="Applicabilityref[Value]=suuid[Value]"
Description="check that all referenced ConceptTemplates in applicability exists
in the mvdxml file" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
      <ConceptRoot uuid="087aae2f-2c43-4f1f-a572-42d0eb27ce2d" name="Existence
of referred ConceptTemplates of Concept in the mvdXML file" applicable-
RootEntity="mvdXML">
        <Concepts>
          <Concept uuid="9de4f802-1301-4b16-87c8-7de4166280af" name="check that
all referenced ConceptTemplates in concept exists in the mvdxml file">
            <Template ref="58c7bea0-2ff8-41f3-97a4-be9854616a94" />
            <Requirements>
              <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules operator="or">
```

```
                <TemplateRule Parameters="Conceptref[Value]=uuid[Value]" Descrip-
tion="check that all referenced ConceptTemplates in concept exists in the mvdxml
file" />
                <TemplateRule Parameters="Conceptref[Value]=suuid[Value]" De-
scription="check that all referenced ConceptTemplates in concept exists in the
mvdxml file" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="087aae2f-2c43-4f1f-a572-42d0eb27ce2e" name="Existence
of referred ConceptTemplates of EntityRule in the mvdXML file" applicable-
RootEntity="mvdXML">
          <Applicability>
            <Template ref="58c7bea0-2ff8-41f3-97a4-be9854616a94" />
            <TemplateRules>
              <TemplateRule Parameters="References[Exists]=TRUE" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="9de4f802-1301-4b16-87c8-7de4166280ae" name="check that
all referenced ConceptTemplates in concept root exists in the mvdxml file">
              <Template ref="58c7bea0-2ff8-41f3-97a4-be9854616a94" />
              <Requirements>
                <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="1496d96f-f4e2-400a-8f75-
c044c7dd6878" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules operator="or">
                <TemplateRule Parameters="Entityref[Value]=uuid[Value]" Descrip-
tion="check that all referenced ConceptTemplates in concept root exists in the
mvdxml file" />
                <TemplateRule Parameters="Entityref[Value]=suuid[Value]" Descrip-
tion="check that all referenced ConceptTemplates in concept root exists in the
mvdxml file" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.8 Constraints

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="efa19fc4-d99b-4762-873a-
02f614f177b4" name="Constraints check mvdXML" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="8e900bab-2693-4a44-88ac-5ba07a654ae7" name="Con-
straints ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="mvdXML">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Templates">
          <EntityRules>
            <EntityRule EntityName="ConceptTemplate">
              <AttributeRules>
                <AttributeRule AttributeName="SubTemplates">
                  <EntityRules>
                    <EntityRule EntityName="ConceptTemplate">
                      <AttributeRules>
                        <AttributeRule AttributeName="Rules">
                          <EntityRules>
                            <EntityRule EntityName="AttributeRule">
                              <AttributeRules>
                                <AttributeRule AttributeName="EntityRules">
                                  <EntityRules>
                                    <EntityRule EntityName="AttributeRuleEnti-
tyRules">
                                      <AttributeRules>
                                        <AttributeRule AttributeName="Enti-
tyRule">
                                          <EntityRules>
                                            <EntityRule EntityName="EntityRule">
                                              <AttributeRules>
                                                <AttributeRule Attribu-
teName="Constraints">
                                                  <EntityRules>
                                                    <EntityRule EntityName="Enti-
tyRuleConstraints">
                                                      <AttributeRules>
                                                        <AttributeRule Attribu-
teName="Constraint" RuleID="SConstraint">
                                                          <EntityRules>
                                                            <EntityRule Enti-
tyName="EntityRuleConstraintsConstraint" />
                                                          </EntityRules>
                                                        </AttributeRule>
                                                      </AttributeRules>
                                                    </EntityRule>
                                                  </EntityRules>
                                                </AttributeRule>
                                              </AttributeRules>
                                            </EntityRule>
                                          </EntityRules>
                                        </AttributeRule>
                                      </AttributeRules>
                                    </EntityRule>
```

```
                                    </EntityRules>
                                  </AttributeRule>
                                </AttributeRules>
                              </EntityRule>
                            </EntityRules>
                          </AttributeRule>
                        </AttributeRules>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                  <AttributeRule AttributeName="Rules">
                    <EntityRules>
                      <EntityRule EntityName="AttributeRule">
                        <AttributeRules>
                          <AttributeRule AttributeName="EntityRules">
                            <EntityRules>
                              <EntityRule EntityName="AttributeRuleEntityRules">
                                <AttributeRules>
                                  <AttributeRule AttributeName="EntityRule">
                                    <EntityRules>
                                      <EntityRule EntityName="EntityRule">
                                        <AttributeRules>
                                          <AttributeRule AttributeName="Con-
straints">
                                            <EntityRules>
                                              <EntityRule EntityName="EntityRule-
Constraints">
                                                <AttributeRules>
                                                  <AttributeRule Attribu-
teName="Constraint" RuleID="Constraint">
                                                    <EntityRules>
                                                      <EntityRule EntityName="Enti-
tyRuleConstraintsConstraint" />
                                                    </EntityRules>
                                                  </AttributeRule>
                                                </AttributeRules>
                                              </EntityRule>
                                            </EntityRules>
                                          </AttributeRule>
                                        </AttributeRules>
                                      </EntityRule>
                                    </EntityRules>
                                  </AttributeRule>
                                </AttributeRules>
                              </EntityRule>
                            </EntityRules>
                          </AttributeRule>
                        </AttributeRules>
                      </EntityRule>
                    </EntityRules>
                  </AttributeRule>
                </AttributeRules>
              </EntityRule>
            </EntityRules>
          </AttributeRule>
          <AttributeRule AttributeName="Views">
            <EntityRules>
              <EntityRule EntityName="ModelView">
                <AttributeRules>
                  <AttributeRule AttributeName="Roots">
                    <EntityRules>
                      <EntityRule EntityName="ConceptRoot">
                        <AttributeRules>
                          <AttributeRule AttributeName="Applicability">
```

```
                        <EntityRules>
                          <EntityRule EntityName="ConceptRootApplicability">
                            <AttributeRules>
                              <AttributeRule AttributeName="TemplateRules">
                                <EntityRules>
                                  <EntityRule EntityName="TemplateRules">
                                    <References IdPrefix="A1_">
                                      <Template ref="2dbb40c4-1433-42f2-9e02-
5216142de6b4" />
                                    </References>
                                    <AttributeRules>
                                      <AttributeRule AttributeName="Items">
                                        <EntityRules>
                                          <EntityRule EntityName="Templat-
eRules">
                                            <References IdPrefix="A2_">
                                              <Template ref="2dbb40c4-1433-
42f2-9e02-5216142de6b4" />
                                            </References>
                                            <AttributeRules>
                                              <AttributeRule Attribu-
teName="Items">
                                                <EntityRules>
                                                  <EntityRule EntityName="Tem-
plateRules">
                                                    <References IdPrefix="A3_">
                                                      <Template ref="2dbb40c4-
1433-42f2-9e02-5216142de6b4" />
                                                    </References>
                                                    <AttributeRules>
                                                      <AttributeRule Attribu-
teName="Items">
                                                        <EntityRules>
                                                          <EntityRule Enti-
tyName="TemplateRules">
                                                            <References IdPre-
fix="A4_">
                                                              <Template
ref="2dbb40c4-1433-42f2-9e02-5216142de6b4" />
                                                            </References>
                                                          </EntityRule>
                                                        </EntityRules>
                                                      </AttributeRule>
                                                    </AttributeRules>
                                                  </EntityRule>
                                                </EntityRules>
                                              </AttributeRule>
                                            </AttributeRules>
                                          </EntityRule>
                                        </EntityRules>
                                      </AttributeRule>
                                    </AttributeRules>
                                  </EntityRule>
                                </EntityRules>
                              </AttributeRule>
                            </AttributeRules>
                          </EntityRule>
                        </EntityRules>
                      </AttributeRule>
                      <AttributeRule AttributeName="Concepts">
                        <EntityRules>
                          <EntityRule EntityName="Concept">
                            <AttributeRules>
                              <AttributeRule AttributeName="TemplateRules">
```

```
                                <EntityRules>
                                  <EntityRule EntityName="TemplateRules">
                                    <References IdPrefix="C1_">
                                      <Template ref="2dbb40c4-1433-42f2-9e02-
5216142de6b4" />
                                    </References>
                                    <AttributeRules>
                                      <AttributeRule AttributeName="Items">
                                        <EntityRules>
                                          <EntityRule EntityName="Templat-
eRules">
                                            <References IdPrefix="C2_">
                                              <Template ref="2dbb40c4-1433-
42f2-9e02-5216142de6b4" />
                                            </References>
                                            <AttributeRules>
                                              <AttributeRule Attribu-
teName="Items">
                                                <EntityRules>
                                                  <EntityRule EntityName="Tem-
plateRules">
                                                    <References IdPrefix="C3_">
                                                      <Template ref="2dbb40c4-
1433-42f2-9e02-5216142de6b4" />
                                                    </References>
                                                    <AttributeRules>
                                                      <AttributeRule Attribu-
teName="Items">
                                                        <EntityRules>
                                                          <EntityRule Enti-
tyName="TemplateRules">
                                                            <References IdPre-
fix="C4_">
                                                              <Template
ref="2dbb40c4-1433-42f2-9e02-5216142de6b4" />
                                                            </References>
                                                          </EntityRule>
                                                        </EntityRules>
                                                      </AttributeRule>
                                                    </AttributeRules>
                                                  </EntityRule>
                                                </EntityRules>
                                              </AttributeRule>
                                            </AttributeRules>
                                          </EntityRule>
                                        </EntityRules>
                                      </AttributeRule>
                                    </AttributeRules>
                                  </EntityRule>
                                </EntityRules>
                              </AttributeRule>
                            </AttributeRules>
                          </EntityRule>
                        </EntityRules>
                      </AttributeRule>
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
```

```
    </ConceptTemplate>
    <ConceptTemplate uuid="2dbb40c4-1433-42f2-9e02-5216142de6b4" name="Templat-
eRules ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Templat-
eRules">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="operator" RuleID="operator" />
        <AttributeRule AttributeName="Items">
          <EntityRules>
            <EntityRule EntityName="TemplateRulesTemplateRule">
              <AttributeRules>
                <AttributeRule AttributeName="Parameters" RuleID="Parameters" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="37d669a7-9e7d-4823-bf75-0ede48ec05eb" name="Constraints Mod-
elView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="c5e9d969-adae-4c05-b6d1-
646ee8cac94f" name="Constraints check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="12e23e60-73af-48c3-ba5b-889fc9ea88b2" name="Con-
straints exist if Applicability parameters is empty" applicable-
RootEntity="mvdXML">
          <Applicability>
            <Template ref="8e900bab-2693-4a44-88ac-5ba07a654ae7" />
            <TemplateRules>
              <TemplateRule Parameters="A1_Parameters[Value]='' OR A2_Parame-
ters[Value]='' OR A3_Parameters[Value]='' OR A4_Parameters[Value]=''" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="3bb3ea16-af65-4806-9f42-4c53464bfece" name="Con-
straints exist if Applicability parameters is empty">
              <Template ref="8e900bab-2693-4a44-88ac-5ba07a654ae7" />
              <TemplateRules operator="or">
                <TemplateRule Parameters="Constraint[Exists]=TRUE" Descrip-
tion="check the existence of a constraint" />
                <TemplateRule Parameters="SConstraint[Exists]=TRUE" Descrip-
tion="check the existence of a constraint" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="12e23e60-73af-48c3-ba5b-889fc9ea88b1" name="Con-
straints exist if Concept parameters is empty" applicableRootEntity="mvdXML">
          <Applicability>
            <Template ref="8e900bab-2693-4a44-88ac-5ba07a654ae7" />
            <TemplateRules>
              <TemplateRule Parameters="C1_Parameters[Value]='' OR C2_Parame-
ters[Value]='' OR C3_Parameters[Value]='' OR C4_Parameters[Value]=''" />
            </TemplateRules>
          </Applicability>
          <Concepts>
```

```
            <Concept uuid="3bb3ea16-af65-4806-9f42-4c53464bfecd" name="Con-
straints exist if Concept parameters is empty">
              <Template ref="8e900bab-2693-4a44-88ac-5ba07a654ae7" />
              <Requirements>
                <Requirement exchangeRequirement="c5e9d969-adae-4c05-b6d1-
646ee8cac94f" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="c5e9d969-adae-4c05-b6d1-
646ee8cac94f" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules operator="or">
                <TemplateRule Parameters="Constraint[Exists]=TRUE" Descrip-
tion="check the existence of a constraint" />
                <TemplateRule Parameters="SConstraint[Exists]=TRUE" Descrip-
tion="check the existence of a constraint" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.9 Name

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="95282cea-9813-40f4-8df1-
09c47baabaa1" name="Name Check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="46ae7cad-d482-4d05-ac12-7b25a64dd2bd" name="mvdXML
name ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="mvdXML">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="name" RuleID="name" />
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="cf07e82c-07b0-4ba1-9b93-5afc3a655438" name="Con-
ceptTemplate name ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="ConceptTemplate">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="name" RuleID="name" />
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="f1199fa9-9558-4953-acd4-3c849ca269bc" name="ModelView
name ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Mod-
elView">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="name" RuleID="name" />
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="54a74e90-2315-4f17-9532-38b8b61baca4" name="Concep-
tRoot name ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Con-
ceptRoot">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="name" RuleID="name" />
      </Rules>
    </ConceptTemplate>
    <ConceptTemplate uuid="085356f4-eb8a-4383-844a-b1fa7667355a" name="Concept
name ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Concept">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
```

```
      <Rules>
        <AttributeRule AttributeName="name" RuleID="name" />
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="b0a45fb3-abcb-42a5-a3fb-2d4c71e90a18" name="Name ModelView"
applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" name="" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="3a205731-14eb-49a0-8749-f6611ef29262" name="Name ex-
istence" applicableRootEntity="mvdXML">
          <Concepts>
            <Concept uuid="b6903a8f-288d-4844-9143-ae58e804fa29" name="Name ex-
istence">
              <Template ref="46ae7cad-d482-4d05-ac12-7b25a64dd2bd" />
              <Requirements>
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="name[Value]!=''" Description="check the
existence of thre required name attribute" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="b34f9bc3-bd00-4dd8-8dd0-9f998a76d0b2" name="Name ex-
istence" applicableRootEntity="ConceptTemplate">
          <Concepts>
            <Concept uuid="b5d030d4-3b85-478a-9244-1cd554a0b9ec" name="Name ex-
istence">
              <Template ref="cf07e82c-07b0-4ba1-9b93-5afc3a655438" />
              <Requirements>
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="name[Value]!=''" Description="check the
existence of thre required name attribute" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="e254aaaf-8e3b-4dc5-95e8-8f407292edc9" name="Name ex-
istence" applicableRootEntity="ModelView">
          <Concepts>
            <Concept uuid="175dca75-39db-4cff-82a8-f55005991d96" name="Name ex-
istence">
              <Template ref="f1199fa9-9558-4953-acd4-3c849ca269bc" />
              <Requirements>
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
```

```
                <TemplateRule Parameters="name[Value]!=''" Description="check the
existence of thre required name attribute" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="e866d8e7-ea66-48a2-bbeb-2d4abe891b27" name="Name ex-
istence" applicableRootEntity="ConceptRoot">
          <Concepts>
            <Concept uuid="d5c61d91-f1fe-4c9a-8d94-4b27394f97d4" name="Name ex-
istence">
              <Template ref="54a74e90-2315-4f17-9532-38b8b61baca4" />
              <Requirements>
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="name[Value]!=''" Description="check the
existence of thre required name attribute" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="f811fd26-6017-4e67-8416-8fa6a3b1a590" name="Name ex-
istence" applicableRootEntity="Concept">
          <Concepts>
            <Concept uuid="8af72edb-a767-44f1-9b7b-bc5f2bc345f5" name="Name ex-
istence">
              <Template ref="085356f4-eb8a-4383-844a-b1fa7667355a" />
              <Requirements>
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="4db6be29-c08f-436c-a8b9-
bd8480b0cf29" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="name[Value]!=''" Description="check the
existence of thre required name attribute" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.10 EntityRule

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="18da6b90-3231-494f-8ee9-
0e07a45eced0" name="EntityRule check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="809c75a7-2fe6-4343-8ccb-ca99e346195f" name="Enti-
tyRuleConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Attribut-
eRule">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="EntityRules">
          <EntityRules>
            <EntityRule EntityName="AttributeRuleEntityRules">
              <AttributeRules>
                <AttributeRule AttributeName="EntityRule" RuleID="EntityRule" />
              </AttributeRules>
            </EntityRule>
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="eed31656-c5e0-486e-8d50-590e7f54a7d1" name="EntityRuleMod-
elView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="17b6e550-0b34-4984-bafe-
d5b114ee496f" name="EntityRule check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="e5a1e8ba-6309-4a04-abb5-648883b691c8" name="Existence
of EntityRule" applicableRootEntity="AttributeRule">
          <Concepts>
            <Concept uuid="5edda7d5-82c9-441f-9f5c-b18d79d91826" name="Existence
of EntityRule">
              <Template ref="809c75a7-2fe6-4343-8ccb-ca99e346195f" />
              <Requirements>
                <Requirement exchangeRequirement="17b6e550-0b34-4984-bafe-
d5b114ee496f" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="17b6e550-0b34-4984-bafe-
d5b114ee496f" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="EntityRule[Exists]=TRUE" Descrip-
tion="check if EntityRule exists" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
      </Roots>
    </ModelView>
  </Views>
</mvdXML>
```

B.11 OperatorExistence

```
<?xml version="1.0" encoding="utf-8"?>
<mvdXML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uuid="90a47549-9b35-4afc-b701-
16570d42c83f" name="OperatorExistence check" xmlns="http://buildingsmart-
tech.org/mvd/XML/1.1">
  <Templates>
    <ConceptTemplate uuid="29b68639-c8ad-4d94-a567-49938a8c730c" name="Operator
ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="ConceptRoot">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="Applicability">
          <EntityRules>
            <EntityRule EntityName="ConceptRootApplicability">
              <AttributeRules>
                <AttributeRule AttributeName="TemplateRules">
                  <EntityRules>
                    <EntityRule EntityName="TemplateRules">
                      <References IdPrefix="A1_">
                        <Template ref="9d3ebda0-6a01-42c2-9463-b641413f3a3d" />
                      </References>
                      <AttributeRules>
                        <AttributeRule AttributeName="Items">
                          <EntityRules>
                            <EntityRule EntityName="TemplateRules">
                              <References IdPrefix="A2_">
                                <Template ref="9d3ebda0-6a01-42c2-9463-
b641413f3a3d" />
                              </References>
                              <AttributeRules>
                                <AttributeRule AttributeName="Items">
                                  <EntityRules>
                                    <EntityRule EntityName="TemplateRules">
                                      <References IdPrefix="A3_">
                                        <Template ref="9d3ebda0-6a01-42c2-9463-
b641413f3a3d" />
                                      </References>
                                      <AttributeRules>
                                        <AttributeRule AttributeName="Items">
                                          <EntityRules>
                                            <EntityRule EntityName="Templat-
eRules">
                                              <References IdPrefix="A4_">
                                                <Template ref="9d3ebda0-6a01-
42c2-9463-b641413f3a3d" />
                                              </References>
                                            </EntityRule>
                                          </EntityRules>
                                        </AttributeRule>
                                      </AttributeRules>
                                    </EntityRule>
                                  </EntityRules>
                                </AttributeRule>
                              </AttributeRules>
                            </EntityRule>
                          </EntityRules>
                        </AttributeRule>
```

```
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
  </ConceptTemplate>
  <ConceptTemplate uuid="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" name="Tem-
plateReference ConceptTemplate" applicableSchema="mvdXML_V1.1" applica-
bleEntity="Concept">
    <Definitions>
      <Definition>
        <Body />
      </Definition>
    </Definitions>
    <Rules>
      <AttributeRule AttributeName="TemplateRules">
        <EntityRules>
          <EntityRule EntityName="TemplateRules">
            <References IdPrefix="C1_">
              <Template ref="9d3ebda0-6a01-42c2-9463-b641413f3a3d" />
            </References>
            <AttributeRules>
              <AttributeRule AttributeName="Items">
                <EntityRules>
                  <EntityRule EntityName="TemplateRules">
                    <References IdPrefix="C2_">
                      <Template ref="9d3ebda0-6a01-42c2-9463-b641413f3a3d" />
                    </References>
                    <AttributeRules>
                      <AttributeRule AttributeName="Items">
                        <EntityRules>
                          <EntityRule EntityName="TemplateRules">
                            <References IdPrefix="C3_">
                              <Template ref="9d3ebda0-6a01-42c2-9463-
b641413f3a3d" />
                            </References>
                            <AttributeRules>
                              <AttributeRule AttributeName="Items">
                                <EntityRules>
                                  <EntityRule EntityName="TemplateRules">
                                    <References IdPrefix="C4_">
                                      <Template ref="9d3ebda0-6a01-42c2-9463-
b641413f3a3d" />
                                    </References>
                                  </EntityRule>
                                </EntityRules>
                              </AttributeRule>
                            </AttributeRules>
                          </EntityRule>
                        </EntityRules>
                      </AttributeRule>
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
```

```
    </ConceptTemplate>
    <ConceptTemplate uuid="9d3ebda0-6a01-42c2-9463-b641413f3a3d" name="Templat-
eRules ConceptTemplate" applicableSchema="mvdXML_V1.1" applicableEntity="Templat-
eRules">
      <Definitions>
        <Definition>
          <Body />
        </Definition>
      </Definitions>
      <Rules>
        <AttributeRule AttributeName="operator" RuleID="Operator" />
        <AttributeRule AttributeName="Items" RuleID="TemplateRule">
          <EntityRules>
            <EntityRule EntityName="object" />
          </EntityRules>
        </AttributeRule>
      </Rules>
    </ConceptTemplate>
  </Templates>
  <Views>
    <ModelView uuid="368715b1-6e25-4bae-9985-5bfa95a00adb" name="OperatorExist-
enceModelView" applicableSchema="mvdXML_V1.1">
      <ExchangeRequirements>
        <ExchangeRequirement applicability="both" uuid="435f8e1c-ee0a-462f-9f48-
b968fb202485" name="OperatorExistence check" />
      </ExchangeRequirements>
      <Roots>
        <ConceptRoot uuid="e000882d-911a-44a8-b6b5-dbbc0c48486d" name="Operator
existence for applicability first TemplatRules" applicableRootEntity="Concep-
tRoot">
          <Applicability>
            <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
            <TemplateRules>
              <TemplateRule Parameters="A1_TemplateRule[Size]&gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="9c9147e3-65f8-4f2e-8d57-817ae1e5e589" name="Operator
existence for applicability first TemplatRules">
              <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="A1_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in first TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="546074ba-6d89-4b23-80ee-4298dffbc479" name="Operator
existence for applicability second TemplatRules" applicableRootEntity="Concep-
tRoot">
          <Applicability>
            <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
            <TemplateRules>
              <TemplateRule Parameters="A2_TemplateRule[Size]&gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
```

```
            <Concept uuid="a60e8004-f2f2-4611-a960-f25e0efe64ea" name="Operator
existence for applicability second TemplatRules">
              <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="A2_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in second TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="e0412f8f-0d85-472d-8c59-c9c7c97dfbcc" name="Operator
existence for applicability third TemplatRules" applicableRootEntity="Concep-
tRoot">
          <Applicability>
            <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
            <TemplateRules>
              <TemplateRule Parameters="A3_TemplateRule[Size]&amp;gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="627d1edd-cd3d-482a-9624-36bcb3c147a5" name="Operator
existence for applicability third TemplatRules">
              <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="A3_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in third TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="29d04bca-962f-495d-b2e4-6bb6531a9b29" name="Operator
existence for applicability fourth TemplatRules" applicableRootEntity="Concep-
tRoot">
          <Applicability>
            <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
            <TemplateRules>
              <TemplateRule Parameters="A4_TemplateRule[Size]&amp;gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="01d1fff8-86a9-4549-83ec-82d7f774e22e" name="Operator
existence for applicability fourth TemplatRules">
              <Template ref="29b68639-c8ad-4d94-a567-49938a8c730c" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
```

```
                <TemplateRule Parameters="A4_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in fourth TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="a42a6025-a02a-46b7-99fb-2064dc7bd435" name="Operator
existence for concept first TemplatRules" applicableRootEntity="Concept">
          <Applicability>
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
            <TemplateRules>
              <TemplateRule Parameters="C1_TemplateRule[Size]&amp;gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="91c11058-e819-40ba-8171-1d5e4ea72545" name="Operator
existence for concept first TemplatRules">
              <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="C1_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in first TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="071ffae7-28e6-4e8a-9668-b297eef53595" name="Operator
existence for concept second TemplatRules" applicableRootEntity="Concept">
          <Applicability>
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
            <TemplateRules>
              <TemplateRule Parameters="C2_TemplateRule[Size]&amp;gt;1" />
            </TemplateRules>
          </Applicability>
          <Concepts>
            <Concept uuid="aec85996-6c68-4ce1-9436-9e1ad799281b" name="Operator
existence for concept second TemplatRules">
              <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
              <Requirements>
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
                <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
              </Requirements>
              <TemplateRules>
                <TemplateRule Parameters="C2_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in second TemplateRules node" />
              </TemplateRules>
            </Concept>
          </Concepts>
        </ConceptRoot>
        <ConceptRoot uuid="216170c7-c249-4b5f-be7c-b8ffb61f9de4" name="Operator
existence for concept third TemplatRules" applicableRootEntity="Concept">
          <Applicability>
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
            <TemplateRules>
```

```
            <TemplateRule Parameters="C3_TemplateRule[Size]&gt;1" />
          </TemplateRules>
        </Applicability>
        <Concepts>
          <Concept uuid="7f640a2d-8922-46cd-908b-d5b1cf08e152" name="Operator
existence for concept third TemplatRules">
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
            <Requirements>
              <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="C3_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in third TemplateRules node" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
      <ConceptRoot uuid="9ced90e7-4d5f-4ddd-839c-0c6ead235a73" name="Operator
existence for concept fourth TemplatRules" applicableRootEntity="Concept">
        <Applicability>
          <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
          <TemplateRules>
            <TemplateRule Parameters="C4_TemplateRule[Size]&gt;1" />
          </TemplateRules>
        </Applicability>
        <Concepts>
          <Concept uuid="4ac0350b-58de-4073-8b99-870c0cebf49d" name="Operator
existence for concept fourth TemplatRules">
            <Template ref="ad9aeda2-ad85-431b-a95b-7ea56fb52b99" />
            <Requirements>
              <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="import" />
              <Requirement exchangeRequirement="435f8e1c-ee0a-462f-9f48-
b968fb202485" requirement="mandatory" applicability="export" />
            </Requirements>
            <TemplateRules>
              <TemplateRule Parameters="C4_Operator[Exists]=TRUE" Descrip-
tion="check the existence of operator when there is more than one Templat-
eRule/TemplateRules in fourth TemplateRules node" />
            </TemplateRules>
          </Concept>
        </Concepts>
      </ConceptRoot>
    </Roots>
  </ModelView>
 </Views>
</mvdXML>
```

Appendix C

Reports

C.1 Furniture Dataset XML against Furniture mvdXML

```
# Furniture Dataset test results

## Chair

### ConceptRoot 'Chair HasBackrest, then IsReclining, has Armrests, HasHeadrest'
(uuid=32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 1

The following entities are passing:
- Chair: 3
- Chair: 4

### ConceptRoot 'Chair Production' (uuid=0c7b70bc-d061-4ff7-bb93-daf67c428ac1)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair Structure=steel, then Cushion=faux leather'
(uuid=86f2a7d8-8e90-485a-a700-f3a8d0f0693b)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair length and width' (uuid=be790918-5b48-416c-9dd2-
ccbe1b51f86a)

The following entities are failing:
- Chair: 2
- Chair: 3

The following entities are passing:
- Chair: 1
- Chair: 4
```

```
### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 4 applicable and 0 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 1

The following entities are passing:
- Chair: 2
- Chair: 3
- Chair: 4

### ConceptRoot 'Chair and Armrests Cushion similar' (uuid=81e78ab9-5560-4ce3-a4d6-7cad75f8920b)

There were 3 applicable and 1 not applicable  Chair(s) in the checked file.

The following entities are failing:
- Chair: 3

The following entities are passing:
- Chair: 2
- Chair: 4

### ConceptRoot 'Chair and Leg colors similar' (uuid=6cc834bf-aced-4f7d-9e20-7c6fd09096da)

The following entities are failing:
- Chair: 2
- Chair: 3

The following entities are passing:
- Chair: 1
- Chair: 4

## Leg

### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 0 applicable and 13 not applicable  Leg(s) in the checked file.

### ConceptRoot 'Leg HasWheel, then IsSwiveling' (uuid=74884d5c-d231-4204-aaeb-3a3fbc13049d)

There were 11 applicable and 2 not applicable  Leg(s) in the checked file.

The following entities are failing:
- Leg: 2.1
- Leg: 2.2
- Leg: 2.3
- Leg: 2.4
- Leg: 3.3

The following entities are passing:
- Leg: 1.1
- Leg: 1.2
- Leg: 1.3
- Leg: 4.1
- Leg: 4.2
- Leg: 4.3

## Armrest
```

```
### ConceptRoot 'Component Cushion exists, then Structure exists' (uuid=31adb6df-f558-4594-b9f3-ed7e3e352bd9)

There were 4 applicable and 0 not applicable  Armrest(s) in the checked file.

The following entities are failing:
- Armrest: 3.1

The following entities are passing:
- Armrest: 2.1
- Armrest: 2.2
- Armrest: 4.1

### ConceptRoot 'Armrest StructureCategory is metal, then steel' (uuid=811c5e38-684d-4684-a54d-a5c4cc0c7517)

There were 3 applicable and 1 not applicable Armrest(s) in the checked file.

The following entities are failing:
- Armrest: 2.1

The following entities are passing:
- Armrest: 2.2
- Armrest: 4.1
```

C.2 Furniture mvdXML against mvdXML Checks

RuleIDReport

```
# RuleID check

## AttributeRule

### ConceptRoot 'RuleID existence' (uuid=5d33acef-c6be-4b54-b9b8-a24ec0c4d043)

The following entities are passing:
- AttributeRule: ID
- AttributeRule: HasBackrest
- AttributeRule: IsReclining
- AttributeRule: HasHeadrest
- AttributeRule: Color
- AttributeRule: Length
- AttributeRule: Width
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Cushion
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Armrests
- AttributeRule: Legs
- AttributeRule: Production
- AttributeRule: IsAntique
- AttributeRule: ProductionYear
- AttributeRule: Country
- AttributeRule: Manufacturer
- AttributeRule: Cushion
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: HasWheel
- AttributeRule: IsSwiveling
- AttributeRule: Color
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Cushion
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category

## ConceptTemplate

### ConceptRoot 'RuleID uniqueness' (uuid=112c33e2-366c-4ab8-9491-47f881bbc07e)

The following entities are passing:
- ConceptTemplate: ab8ea1a0-3610-4e2f-b3d6-0f6414522d94
- ConceptTemplate: 5d2e52c2-cc53-441d-99b2-58df8312dda8
- ConceptTemplate: 28edd9a1-fdd2-4298-9a94-49c7ccc031b3
- ConceptTemplate: 9f10223f-0da5-4536-9e17-0f96f13dd6e8
```

C.2 Furniture mvdXML against mvdXML Checks

IdPrefixReport

```
# IdPrefix check

## EntityRule

### ConceptRoot 'IdPrefix existence' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebc)

There were 2 applicable and 8 not applicable  EntityRule(s) in the checked file.

The following entities are passing:
- EntityRule: Armrest
- EntityRule: Leg

## ConceptTemplate

### ConceptRoot 'IdPrefix uniqueness' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebd)

There were 1 applicable and 3 not applicable  ConceptTemplate(s) in the checked
file.

The following entities are passing:
- ConceptTemplate: ab8ea1a0-3610-4e2f-b3d6-0f6414522d94
```

MetricValueReport

```
# MetricValue check

## TemplateRulesTemplateRule

### ConceptRoot 'MetricValue' (uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b0)

The following entities are passing:
- TemplateRulesTemplateRule: HasBackrest[Value]=TRUE
- TemplateRulesTemplateRule: IsReclining[Value]=TRUE AND Armrests[Size]>0 AND
HasHeadrest[Value]=TRUE
- TemplateRulesTemplateRule: IsAntique[Value]=TRUE
- TemplateRulesTemplateRule: Country[Exists]=TRUE AND Manufacturer[Exists]=TRUE
AND ProductionYear[Exists]=TRUE
- TemplateRulesTemplateRule: StructureMaterialName[Value]='Steel'
- TemplateRulesTemplateRule: CushionMaterialName[Value]='Faux Leather'
- TemplateRulesTemplateRule: Length[Value]>=52.5 AND Width[Value]>=48.5
- TemplateRulesTemplateRule: Cushion[Exists]=TRUE
- TemplateRulesTemplateRule: Structure[Exists]=TRUE
- TemplateRulesTemplateRule: HasWheel[Value]=TRUE
- TemplateRulesTemplateRule: IsSwiveling[Value]=TRUE
- TemplateRulesTemplateRule: StructureCategory[Value]='Metal'
- TemplateRulesTemplateRule: StructureMaterialName[Value]='Steel'
- TemplateRulesTemplateRule: Armrests[Size]>0
- TemplateRulesTemplateRule: Armrests_CushionMaterialName[Value]=CushionMaterial-
Name[Value]
- TemplateRulesTemplateRule: Color[Value]=Legs_Color[Value]
```

C.2 Furniture mvdXML against mvdXML Checks

ParametersAttributesReport

```
# ParametersAttributes check

## mvdXML

### ConceptRoot 'Applicability parameters attributes' (uuid=81a939d4-7568-460e-
b212-346efb5d0af4)

The following entities are failing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701

### ConceptRoot 'Concept parameters attributes' (uuid=f1d6e1cd-4007-44ae-aaa9-
ab3960555501)

The following entities are failing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701
```

C.2 Furniture mvdXML against mvdXML Checks

TemplateReferenceReport

```
# TemplateReference check
## ConceptRoot

### ConceptRoot 'Applicability template reference' (uuid=2e84ff23-2825-49e8-a92e-8af75bb92753)

There were 7 applicable and 2 not applicable ConceptRoot(s) in the checked file.

The following entities are passing:
- ConceptRoot: 32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7
- ConceptRoot: 0c7b70bc-d061-4ff7-bb93-daf67c428ac1
- ConceptRoot: 86f2a7d8-8e90-485a-a700-f3a8d0f0693b
- ConceptRoot: 31adb6df-f558-4594-b9f3-ed7e3e352bd9
- ConceptRoot: 74884d5c-d231-4204-aaeb-3a3fbc13049d
- ConceptRoot: 811c5e38-684d-4684-a54d-a5c4cc0c7517
- ConceptRoot: 81e78ab9-5560-4ce3-a4d6-7cad75f8920b

## Concept

### ConceptRoot 'Concept template reference' (uuid=d80f9f4a-19f5-4585-aaed-1fed551ad8f7)

The following entities are passing:
- Concept: fdc7fec5-6e23-4e99-b3e0-c84832662a6c
- Concept: 2300768b-1297-4bb8-b348-978c449633b7
- Concept: cf38f321-63cc-4c15-93e3-09291441552d
- Concept: 3aa515cf-c928-4f8a-8fb3-89b252c5e2b1
- Concept: afe6bbcc-91a8-477f-95ff-318a2c832e0c
- Concept: 2c5eaeec-cc64-45da-9cc6-e2f7257f59d1
- Concept: 9ba7e275-5012-43a0-a63c-714c8b5065ee
- Concept: ac7f9692-913f-4992-b276-b3b9a3bddb0d
- Concept: 99c1cc18-ef3b-4ef6-9049-58e8d4ac14ef

## EntityRule

### ConceptRoot 'Entity template reference' (uuid=44401324-8c79-4cfc-b40d-29a105f4de61)

There were 2 applicable and 8 not applicable  EntityRule(s) in the checked file.

The following entities are passing:
- EntityRule: Armrest
- EntityRule: Leg
```

EntityReferenceReport

```
### ConceptRoot 'Entity Reference' (uuid=44401324-8c79-4cfc-b40d-29a105f4de69)

The following entities are failing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701
```

C.2 Furniture mvdXML against mvdXML Checks

ReferencedConceptTemplatesReport

```
# ReferencedConceptTemplates check

## mvdXML

### ConceptRoot 'Existence of referred ConceptTemplates check of Applicability in
the mvdXML file' (uuid=b3d73a9a-3693-462a-bed6-20658ed48122)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are passing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701

### ConceptRoot 'Existence of referred ConceptTemplates of Concept in the mvdXML
file' (uuid=087aae2f-2c43-4f1f-a572-42d0eb27ce2d)

The following entities are passing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701

### ConceptRoot 'Existence of referred ConceptTemplates of EntityRule in the
mvdXML file' (uuid=087aae2f-2c43-4f1f-a572-42d0eb27ce2e)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are failing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701
```

ConstraintsReport

```
# Constraints check

## mvdXML

### ConceptRoot 'Constraints exist if Applicability parameters is empty'
(uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b2)

There were 0 applicable and 1 not applicable mvdXML(s) in the checked file.

### ConceptRoot 'Constraints exist if Concept parameters is empty'
(uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b1)

There were 0 applicable and 1 not applicable mvdXML(s) in the checked file.
```

C.2 Furniture mvdXML against mvdXML Checks

NameReport

```
# Name check

## mvdXML

### ConceptRoot 'Name existence' (uuid=3a205731-14eb-49a0-8749-f6611ef29262)

The following entities are passing:
- mvdXML: 9ba97c1d-c13c-4b4f-9581-51565828e701

## ConceptTemplate

### ConceptRoot 'Name existence' (uuid=b34f9bc3-bd00-4dd8-8dd0-9f998a76d0b2)

The following entities are passing:
- ConceptTemplate: ab8ea1a0-3610-4e2f-b3d6-0f6414522d94
- ConceptTemplate: 5d2e52c2-cc53-441d-99b2-58df8312dda8
- ConceptTemplate: 28edd9a1-fdd2-4298-9a94-49c7ccc031b3
- ConceptTemplate: 9f10223f-0da5-4536-9e17-0f96f13dd6e8

## ModelView

### ConceptRoot 'Name existence' (uuid=e254aaaf-8e3b-4dc5-95e8-8f407292edc9)

The following entities are passing:
- ModelView: 83f5229e-f411-466e-b25b-b89d04666e1a

## ConceptRoot

### ConceptRoot 'Name existence' (uuid=e866d8e7-ea66-48a2-bbeb-2d4abe891b27)

The following entities are passing:
- ConceptRoot: 32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7
- ConceptRoot: 0c7b70bc-d061-4ff7-bb93-daf67c428ac1
- ConceptRoot: 86f2a7d8-8e90-485a-a700-f3a8d0f0693b
- ConceptRoot: be790918-5b48-416c-9dd2-ccbe1b51f86a
- ConceptRoot: 31adb6df-f558-4594-b9f3-ed7e3e352bd9
- ConceptRoot: 74884d5c-d231-4204-aaeb-3a3fbc13049d
- ConceptRoot: 811c5e38-684d-4684-a54d-a5c4cc0c7517
- ConceptRoot: 81e78ab9-5560-4ce3-a4d6-7cad75f8920b
- ConceptRoot: 6cc834bf-aced-4f7d-9e20-7c6fd09096da

## Concept

### ConceptRoot 'Name existence' (uuid=f811fd26-6017-4e67-8416-8fa6a3b1a590)

The following entities are passing:
- Concept: fdc7fec5-6e23-4e99-b3e0-c84832662a6c
- Concept: 2300768b-1297-4bb8-b348-978c449633b7
- Concept: cf38f321-63cc-4c15-93e3-09291441552d
- Concept: 3aa515cf-c928-4f8a-8fb3-89b252c5e2b1
- Concept: afe6bbcc-91a8-477f-95ff-318a2c832e0c
- Concept: 2c5eaeec-cc64-45da-9cc6-e2f7257f59d1
- Concept: 9ba7e275-5012-43a0-a63c-714c8b5065ee
- Concept: ac7f9692-913f-4992-b276-b3b9a3bddb0d
- Concept: 99c1cc18-ef3b-4ef6-9049-58e8d4ac14ef
```

C.2 Furniture mvdXML against mvdXML Checks

EntityRuleReport

```
# EntityRule check

## AttributeRule

### ConceptRoot 'Existence of EntityRule' (uuid=e5a1e8ba-6309-4a04-abb5-
648883b691c8)

The following entities are failing:
- AttributeRule: ID
- AttributeRule: HasBackrest
- AttributeRule: IsReclining
- AttributeRule: HasHeadrest
- AttributeRule: Color
- AttributeRule: Length
- AttributeRule: Width
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: IsAntique
- AttributeRule: ProductionYear
- AttributeRule: Country
- AttributeRule: Manufacturer
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: HasWheel
- AttributeRule: IsSwiveling
- AttributeRule: Color
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Name
- AttributeRule: Category

The following entities are passing:
- AttributeRule: Structure
- AttributeRule: Cushion
- AttributeRule: Armrests
- AttributeRule: Legs
- AttributeRule: Production
- AttributeRule: Cushion
- AttributeRule: Structure
- AttributeRule: Structure
- AttributeRule: Cushion
- AttributeRule: Structure
```

C.2 Furniture mvdXML against mvdXML Checks

OperatorExistenceReport

```
# OperatorExistence check

## ConceptRoot

### ConceptRoot 'Operator existence for applicability first TemplatRules'
(uuid=e000882d-911a-44a8-b6b5-dbbc0c48486d)

There were 0 applicable and 9 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability second TemplatRules'
(uuid=546074ba-6d89-4b23-80ee-4298dffbc479)

There were 0 applicable and 9 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability third TemplatRules'
(uuid=e0412f8f-0d85-472d-8c59-c9c7c97dfbcc)

There were 0 applicable and 9 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability fourth TemplatRules'
(uuid=29d04bca-962f-495d-b2e4-6bb6531a9b29)

There were 0 applicable and 9 not applicable ConceptRoot(s) in the checked file.

## Concept

### ConceptRoot 'Operator existence for concept first TemplatRules'
(uuid=a42a6025-a02a-46b7-99fb-2064dc7bd435)

There were 0 applicable and 9 not applicable Concept(s) in the checked file.

### ConceptRoot 'Operator existence for concept second TemplatRules'
(uuid=071ffae7-28e6-4e8a-9668-b297eef53595)

There were 0 applicable and 9 not applicable Concept(s) in the checked file.

### ConceptRoot 'Operator existence for concept third TemplatRules'
(uuid=216170c7-c249-4b5f-be7c-b8ffb61f9de4)

There were 0 applicable and 9 not applicable Concept(s) in the checked file.

### ConceptRoot 'Operator existence for concept fourth TemplatRules'
(uuid=9ced90e7-4d5f-4ddd-839c-0c6ead235a73)

There were 0 applicable and 9 not applicable Concept(s) in the checked file.
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ComponentRuleIDUniquenessReport

```
# ComponentRuleID Uniqueness check

## AttributeRule

### ConceptRoot 'RuleID existence' (uuid=5d33acef-c6be-4b54-b9b8-a24ec0c4d043)

The following entities are passing:
- AttributeRule: Cushion
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category

## ConceptTemplate

### ConceptRoot 'RuleID uniqueness' (uuid=112c33e2-366c-4ab8-9491-47f881bbc07e)

The following entities are failing:
- ConceptTemplate: 9f10223f-0da5-4536-9e17-0f96f13dd6e8
```

ComponentRuleIDExistenceReport

```
# ComponentRuleID Existence check

## AttributeRule

### ConceptRoot 'RuleID existence' (uuid=5d33acef-c6be-4b54-b9b8-a24ec0c4d043)

The following entities are failing:
- AttributeRule: Cushion
- AttributeRule: Name
- AttributeRule: Category
- AttributeRule: Structure
- AttributeRule: Name
- AttributeRule: Category

## ConceptTemplate

### ConceptRoot 'RuleID uniqueness' (uuid=112c33e2-366c-4ab8-9491-47f881bbc07e)

The following entities are failing:
- ConceptTemplate: 9f10223f-0da5-4536-9e17-0f96f13dd6e8
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ChairIdPrefixUniquenessReport

```
# ChairIdPrefixUniqueness check

## EntityRule

### ConceptRoot 'IdPrefix existence' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebc)

There were 2 applicable and 6 not applicable  EntityRule(s) in the checked file.

The following entities are passing:
- EntityRule: Armrest
- EntityRule: Leg

## ConceptTemplate

### ConceptRoot 'IdPrefix uniqueness' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebd)

There were 1 applicable and 2 not applicable  ConceptTemplate(s) in the checked
file.

The following entities are failing:
- ConceptTemplate: ab8ea1a0-3610-4e2f-b3d6-0f6414522d94
```

ChairIdPrefixExistenceReport

```
# ChairIdPrefixExistence check

## EntityRule

### ConceptRoot 'IdPrefix existence' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebc)

There were 2 applicable and 6 not applicable  EntityRule(s) in the checked file.

The following entities are failing:
- EntityRule: Armrest
- EntityRule: Leg

## ConceptTemplate

### ConceptRoot 'IdPrefix uniqueness' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebd)

There were 0 applicable and 3 not applicable  ConceptTemplate(s) in the checked
file.
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ComponentMetricValueReport

```
# ComponentMetricValue check

## TemplateRulesTemplateRule

### ConceptRoot 'MetricValue' (uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b0)

The following entities are failing:
- TemplateRulesTemplateRule: Cushion[Valu]=TRUE
- TemplateRulesTemplateRule: CushionCategory[]='Leather'
- TemplateRulesTemplateRule: Structure[Exists]=TRUE AND StructureCate-
gory[]='Metal'
- TemplateRulesTemplateRule: StructureMaterialName[Exist]=TRUE
- TemplateRulesTemplateRule: Structure[Exists]=TRUE AND StructureCate-
gory[yes]=TRUE
```

ComponentParametersAttributesReport

```
# ComponentParametersAttributes check

## mvdXML

### ConceptRoot 'Applicability parameters attributes' (uuid=81a939d4-7568-460e-
b212-346efb5d0af4)

The following entities are failing:
- mvdXML: a6f819ae-8715-465d-b6a7-d4b76298efbc

### ConceptRoot 'Concept parameters attributes' (uuid=f1d6e1cd-4007-44ae-aaa9-
ab3960555501)

The following entities are failing:
- mvdXML: a6f819ae-8715-465d-b6a7-d4b76298efbc
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ChairTemplateReferenceReport

```
# ChairTemplateReference check

## ConceptRoot

### ConceptRoot 'Applicability template reference' (uuid=2e84ff23-2825-49e8-a92e-8af75bb92753)

There were 1 applicable and 0 not applicable  ConceptRoot(s) in the checked file.

The following entities are failing:
- ConceptRoot: cc841c98-136d-4791-a8cb-ee8204e0bc4b

## Concept

### ConceptRoot 'Concept template reference' (uuid=d80f9f4a-19f5-4585-aaed-1fed551ad8f7)

The following entities are failing:
- Concept: 6d41b0ee-e4e2-46cb-8376-1f025bba0b8f

## EntityRule

### ConceptRoot 'Entity template reference' (uuid=44401324-8c79-4cfc-b40d-29a105f4de61)

There were 2 applicable and 6 not applicable  EntityRule(s) in the checked file.

The following entities are failing:
- EntityRule: Armrest
- EntityRule: Leg
```

Chair-LegEntityReferenceReport

```
# Chair-LegEntityReference check

## mvdXML

### ConceptRoot 'Entity Reference' (uuid=44401324-8c79-4cfc-b40d-29a105f4de69)

The following entities are failing:
- mvdXML: f65063a2-b4b2-4da5-8cb8-bb26d33bd38b
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ChairReferencedConceptTemplatesReport

```
# ChairReferencedConceptTemplates check

## mvdXML

### ConceptRoot 'Existence of referred ConceptTemplates check of Applicability in
the mvdXML file' (uuid=b3d73a9a-3693-462a-bed6-20658ed48122)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are failing:
- mvdXML: 7f06d0ef-1246-41b5-bb10-0b0c12e511d2

### ConceptRoot 'Existence of referred ConceptTemplates of Concept in the mvdXML
file' (uuid=087aae2f-2c43-4f1f-a572-42d0eb27ce2d)

The following entities are failing:
- mvdXML: 7f06d0ef-1246-41b5-bb10-0b0c12e511d2

### ConceptRoot 'Existence of referred ConceptTemplates of EntityRule in the
mvdXML file' (uuid=087aae2f-2c43-4f1f-a572-42d0eb27ce2e)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are failing:
- mvdXML: 7f06d0ef-1246-41b5-bb10-0b0c12e511d2
```

ChairConstraintsReport

```
# ChairConstraints check

## mvdXML

### ConceptRoot 'Constraints exist if Applicability parameters is empty'
(uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b2)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are failing:
- mvdXML: 266ad15b-9dac-46de-b500-bdda6e0e6d79

### ConceptRoot 'Constraints exist if Concept parameters is empty'
(uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b1)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are failing:
- mvdXML: 266ad15b-9dac-46de-b500-bdda6e0e6d79
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ChairNameReport

```
# ChairName check

## mvdXML

### ConceptRoot 'Name existence' (uuid=3a205731-14eb-49a0-8749-f6611ef29262)

The following entities are failing:
- mvdXML: c8a623f4-7d9e-428a-9c56-74e0cdce9923

## ConceptTemplate

### ConceptRoot 'Name existence' (uuid=b34f9bc3-bd00-4dd8-8dd0-9f998a76d0b2)

The following entities are failing:
- ConceptTemplate: ab8ea1a0-3610-4e2f-b3d6-0f6414522d94

## ModelView

### ConceptRoot 'Name existence' (uuid=e254aaaf-8e3b-4dc5-95e8-8f407292edc9)

The following entities are failing:
- ModelView: 6fe45497-9985-495a-a1a9-fa8c64f457b9

## ConceptRoot

### ConceptRoot 'Name existence' (uuid=e866d8e7-ea66-48a2-bbeb-2d4abe891b27)

The following entities are failing:
- ConceptRoot: 32c2ca93-451e-4fc2-9891-ad0dcf6bb3e7

## Concept

### ConceptRoot 'Name existence' (uuid=f811fd26-6017-4e67-8416-8fa6a3b1a590)

The following entities are failing:
- Concept: fdc7fec5-6e23-4e99-b3e0-c84832662a6c
```

C.3 Incorrect Furniture mvdXML files against mvdXML Checks

ChairOperatorExistenceReport

```
# ChairOperatorExistence check

## ConceptRoot

### ConceptRoot 'Operator existence for applicability first TemplatRules'
(uuid=e000882d-911a-44a8-b6b5-dbbc0c48486d)

There were 0 applicable and 1 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability second TemplatRules'
(uuid=546074ba-6d89-4b23-80ee-4298dffbc479)

There were 0 applicable and 1 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability third TemplatRules'
(uuid=e0412f8f-0d85-472d-8c59-c9c7c97dfbcc)

There were 0 applicable and 1 not applicable  ConceptRoot(s) in the checked file.

### ConceptRoot 'Operator existence for applicability fourth TemplatRules'
(uuid=29d04bca-962f-495d-b2e4-6bb6531a9b29)

There were 0 applicable and 1 not applicable  ConceptRoot(s) in the checked file.

## Concept

### ConceptRoot 'Operator existence for concept first TemplatRules'
(uuid=a42a6025-a02a-46b7-99fb-2064dc7bd435)

There were 1 applicable and 1 not applicable  Concept(s) in the checked file.

The following entities are passing:
- Concept: 12ad3dad-264a-4f6e-ac4d-df669d03b9a2

### ConceptRoot 'Operator existence for concept second TemplatRules'
(uuid=071ffae7-28e6-4e8a-9668-b297eef53595)

There were 1 applicable and 1 not applicable Concept(s) in the checked file.

The following entities are passing:
- Concept: 12ad3dad-264a-4f6e-ac4d-df669d03b9a2

### ConceptRoot 'Operator existence for concept third TemplatRules'
(uuid=216170c7-c249-4b5f-be7c-b8ffb61f9de4)

There were 0 applicable and 2 not applicable Concept(s) in the checked file.

### ConceptRoot 'Operator existence for concept fourth TemplatRules'
(uuid=9ced90e7-4d5f-4ddd-839c-0c6ead235a73)

There were 0 applicable and 2 not applicable  Concept(s) in the checked file.
```

C.4 Official mvdXML files and other examples against mvdXML Checks

The reports have huge space and can be found within the enclosed ZIP file.

C.5 mvdXMLChecks file against mvdXMLChecks

```
# MvdXMLChecks

## ConceptTemplate

### ConceptRoot 'RuleID uniqueness' (uuid=112c33e2-366c-4ab8-9491-47f881bbc07e)

The following entities are passing:
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec311
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec312
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02e
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02f
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae8
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b96
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b97
- ConceptTemplate: c9d40546-597e-41a5-9f1f-2f2ef50ca54c
- ConceptTemplate: 58c7bea0-2ff8-41f3-97a4-be9854616a94
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae7
- ConceptTemplate: 2dbb40c4-1433-42f2-9e02-5216142de6b4
- ConceptTemplate: cc727b3c-f7f8-45a5-a91c-6b417dca5015
- ConceptTemplate: 3da5173a-417a-4d03-91c2-e62ad8049cf6
- ConceptTemplate: 8192b792-53ba-4153-a70c-ba92be020070
- ConceptTemplate: 1bd2b0b0-155f-46a4-86a5-f2ad4ab76922
- ConceptTemplate: 52f03d6e-6248-4750-ba15-b0842c1ccf1a
- ConceptTemplate: 809c75a7-2fe6-4343-8ccb-ca99e346195f

### ConceptRoot 'IdPrefix uniqueness' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebd)

There were 17 applicable and 0 not applicable  ConceptTemplate(s) in the checked
file.

The following entities are passing:
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec311
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec312
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02e
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02f
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae8
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b96
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b97
- ConceptTemplate: c9d40546-597e-41a5-9f1f-2f2ef50ca54c
- ConceptTemplate: 58c7bea0-2ff8-41f3-97a4-be9854616a94
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae7
- ConceptTemplate: 2dbb40c4-1433-42f2-9e02-5216142de6b4
- ConceptTemplate: cc727b3c-f7f8-45a5-a91c-6b417dca5015
- ConceptTemplate: 3da5173a-417a-4d03-91c2-e62ad8049cf6
- ConceptTemplate: 8192b792-53ba-4153-a70c-ba92be020070
- ConceptTemplate: 1bd2b0b0-155f-46a4-86a5-f2ad4ab76922
- ConceptTemplate: 52f03d6e-6248-4750-ba15-b0842c1ccf1a
- ConceptTemplate: 809c75a7-2fe6-4343-8ccb-ca99e346195f
```

ConceptRoot 'ConceptTemplate Name existence' (uuid=f823d6e0-ee95-4bb0-a62b-249a17bc283d)

The following entities are passing:
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec311
- ConceptTemplate: dd83efa6-88c4-4954-b378-7f6ffaaec312
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02e
- ConceptTemplate: 4c31b9d9-3a83-4120-a56d-9b116593e02f
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae8
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b96
- ConceptTemplate: ad9aeda2-ad85-431b-a95b-7ea56fb52b97
- ConceptTemplate: c9d40546-597e-41a5-9f1f-2f2ef50ca54c
- ConceptTemplate: 58c7bea0-2ff8-41f3-97a4-be9854616a94
- ConceptTemplate: 8e900bab-2693-4a44-88ac-5ba07a654ae7
- ConceptTemplate: 2dbb40c4-1433-42f2-9e02-5216142de6b4
- ConceptTemplate: cc727b3c-f7f8-45a5-a91c-6b417dca5015
- ConceptTemplate: 3da5173a-417a-4d03-91c2-e62ad8049cf6
- ConceptTemplate: 8192b792-53ba-4153-a70c-ba92be020070
- ConceptTemplate: 1bd2b0b0-155f-46a4-86a5-f2ad4ab76922
- ConceptTemplate: 52f03d6e-6248-4750-ba15-b0842c1ccf1a
- ConceptTemplate: 809c75a7-2fe6-4343-8ccb-ca99e346195f

AttributeRule

ConceptRoot 'RuleID existence' (uuid=5d33acef-c6be-4b54-b9b8-a24ec0c4d043)

The following entities are passing:
- AttributeRule: RuleID
- AttributeRule: Rules
- AttributeRule: References
- AttributeRule: IdPrefix
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Parameters
- AttributeRule: Applicability
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: References
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Views
- AttributeRule: Roots
- AttributeRule: Applicability
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Concepts
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Templates
- AttributeRule: uuid
- AttributeRule: SubTemplates
- AttributeRule: uuid
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: References
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Templates
- AttributeRule: SubTemplates
- AttributeRule: Rules

```
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Constraints
- AttributeRule: Constraint
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Constraints
- AttributeRule: Constraint
- AttributeRule: Views
- AttributeRule: Roots
- AttributeRule: Applicability
- AttributeRule: TemplateRules
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Concepts
- AttributeRule: TemplateRules
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Parameters
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: EntityRules
- AttributeRule: EntityRule

### ConceptRoot 'EntityRule existence' (uuid=e5a1e8ba-6309-4a04-abb5-
648883b691c8)

The following entities are passing:
- AttributeRule: RuleID
- AttributeRule: Rules
- AttributeRule: References
- AttributeRule: IdPrefix
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Parameters
- AttributeRule: Applicability
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: References
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Views
- AttributeRule: Roots
- AttributeRule: Applicability
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Concepts
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Templates
- AttributeRule: uuid
- AttributeRule: SubTemplates
- AttributeRule: uuid
- AttributeRule: Rules
```

```
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: References
- AttributeRule: Template
- AttributeRule: ref
- AttributeRule: Templates
- AttributeRule: SubTemplates
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Constraints
- AttributeRule: Constraint
- AttributeRule: Rules
- AttributeRule: EntityRules
- AttributeRule: EntityRule
- AttributeRule: Constraints
- AttributeRule: Constraint
- AttributeRule: Views
- AttributeRule: Roots
- AttributeRule: Applicability
- AttributeRule: TemplateRules
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Concepts
- AttributeRule: TemplateRules
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Items
- AttributeRule: Parameters
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: name
- AttributeRule: EntityRules
- AttributeRule: EntityRule

## EntityRule

### ConceptRoot 'IdPrefix existence' (uuid=0a48b28a-c7b4-4e53-8153-d6042aaf0ebc)

There were 10 applicable and 57 not applicable EntityRule(s) in the checked file.

The following entities are passing:
- EntityRule: AttributeRule
- EntityRule: EntityRule
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules

### ConceptRoot 'Entity template reference' (uuid=44401324-8c79-4cfc-b40d-
29a105f4de61)

There were 10 applicable and 57 not applicable EntityRule(s) in the checked file.

The following entities are passing:
- EntityRule: AttributeRule
```

```
- EntityRule: EntityRule
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules
- EntityRule: TemplateRules

## TemplateRulesTemplateRule

### ConceptRoot 'MetricValue' (uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b0)

The following entities are passing:
- TemplateRulesTemplateRule: A_RuleID[Unique]=TRUE
- TemplateRulesTemplateRule: RuleID[Exists]=TRUE
- TemplateRulesTemplateRule: References[Exists]=TRUE
- TemplateRulesTemplateRule: IdPrefix[Exists]=TRUE
- TemplateRulesTemplateRule: IdPrefix[Exists]=TRUE
- TemplateRulesTemplateRule: E_IdPrefix[Unique]=TRUE
- TemplateRulesTemplateRule: Parameters[Value]!=reg'.*(\\[\\]).*' AND Parame-
ters[Value]=reg'.*(\\[Value\\]|\\[VALUE\\]|\\[Size\\]|\\[SIZE\\]|\\[Type\\]|\\[TY
PE\\]|\\[Unique\\]|\\[UNIQUE\\]|\\[Exists\\]|\\[EXISTS\\]).*'
- TemplateRulesTemplateRule: Parameters[Value]!=reg'.*(\\[[a-z]+\\]).*' AND Pa-
rame-
ters[Value]=reg'.*(\\[Value\\]|\\[VALUE\\]|\\[Size\\]|\\[SIZE\\]|\\[Type\\]|\\[TY
PE\\]|\\[Unique\\]|\\[UNIQUE\\]|\\[Exists\\]|\\[EXISTS\\]).*'
- TemplateRulesTemplateRule: Applicability[Exists]=TRUE
- TemplateRulesTemplateRule: Applicabilityref[Value]!=''
- TemplateRulesTemplateRule: Conceptref[Value]!=''
- TemplateRulesTemplateRule: References[Exists]=TRUE
- TemplateRulesTemplateRule: ref[Value]!=''
- TemplateRulesTemplateRule: Applicability[Exists]=TRUE
- TemplateRulesTemplateRule: Applicabilityref[Value]=uuid[Value]
- TemplateRulesTemplateRule: Applicabilityref[Value]=suuid[Value]
- TemplateRulesTemplateRule: Conceptref[Value]=uuid[Value]
- TemplateRulesTemplateRule: Conceptref[Value]=suuid[Value]
- TemplateRulesTemplateRule: A1_Parameters[Value]='' OR A2_Parameters[Value]=''
OR A3_Parameters[Value]='' OR A4_Parameters[Value]=''
- TemplateRulesTemplateRule: Constraint[Exists]=TRUE
- TemplateRulesTemplateRule: SConstraint[Exists]=TRUE
- TemplateRulesTemplateRule: C1_Parameters[Value]='' OR C2_Parameters[Value]=''
OR C3_Parameters[Value]='' OR C4_Parameters[Value]=''
- TemplateRulesTemplateRule: Constraint[Exists]=TRUE
- TemplateRulesTemplateRule: SConstraint[Exists]=TRUE
- TemplateRulesTemplateRule: name[Value]!=''
- TemplateRulesTemplateRule: name[Value]!=''
- TemplateRulesTemplateRule: name[Value]!=''
- TemplateRulesTemplateRule: name[Value]!=''
- TemplateRulesTemplateRule: name[Value]!=''
- TemplateRulesTemplateRule: EntityRule[Exists]=TRUE

## ConceptRoot

### ConceptRoot 'Applicability template reference' (uuid=2e84ff23-2825-49e8-a92e-
8af75bb92753)

There were 7 applicable and 11 not applicable  ConceptRoot(s) in the checked
file.

The following entities are passing:
- ConceptRoot: 0a48b28a-c7b4-4e53-8153-d6042aaf0ebc
- ConceptRoot: 0a48b28a-c7b4-4e53-8153-d6042aaf0ebd
```

- ConceptRoot: 2e84ff23-2825-49e8-a92e-8af75bb92753
- ConceptRoot: 44401324-8c79-4cfc-b40d-29a105f4de61
- ConceptRoot: b3d73a9a-3693-462a-bed6-20658ed48122
- ConceptRoot: 12e23e60-73af-48c3-ba5b-889fc9ea88b2
- ConceptRoot: 12e23e60-73af-48c3-ba5b-889fc9ea88b1

ConceptRoot 'ConceptRoot Name existence' (uuid=a188ab71-7624-4b24-81e4-fcc4fb73ee0f)

The following entities are passing:
- ConceptRoot: 112c33e2-366c-4ab8-9491-47f881bbc07e
- ConceptRoot: 5d33acef-c6be-4b54-b9b8-a24ec0c4d043
- ConceptRoot: 0a48b28a-c7b4-4e53-8153-d6042aaf0ebc
- ConceptRoot: 0a48b28a-c7b4-4e53-8153-d6042aaf0ebd
- ConceptRoot: 12e23e60-73af-48c3-ba5b-889fc9ea88b0
- ConceptRoot: 2e84ff23-2825-49e8-a92e-8af75bb92753
- ConceptRoot: d80f9f4a-19f5-4585-aaed-1fed551ad8f7
- ConceptRoot: 44401324-8c79-4cfc-b40d-29a105f4de61
- ConceptRoot: b3d73a9a-3693-462a-bed6-20658ed48122
- ConceptRoot: 087aae2f-2c43-4f1f-a572-42d0eb27ce2d
- ConceptRoot: 12e23e60-73af-48c3-ba5b-889fc9ea88b2
- ConceptRoot: 12e23e60-73af-48c3-ba5b-889fc9ea88b1
- ConceptRoot: fae1d217-26da-4cb3-aa2a-dcc82b059c0b
- ConceptRoot: f823d6e0-ee95-4bb0-a62b-249a17bc283d
- ConceptRoot: c268d637-1674-4948-8eb3-0e185a8536a3
- ConceptRoot: a188ab71-7624-4b24-81e4-fcc4fb73ee0f
- ConceptRoot: f210fea4-f81c-4333-81b9-6912585703fe
- ConceptRoot: e5a1e8ba-6309-4a04-abb5-648883b691c8

Concept

ConceptRoot 'Concept template reference' (uuid=d80f9f4a-19f5-4585-aaed-1fed551ad8f7)

The following entities are passing:
- Concept: e5eaa92e-d530-414f-bf86-c50a3b84fbe2
- Concept: 68ecc19c-cd8c-4a38-b0d2-8bf16d2b6455
- Concept: cb576fee-b25f-4d51-aa07-cd436851317d
- Concept: cb576fee-b25f-4d51-aa07-cd436851317c
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfecc
- Concept: b45128ae-dc95-4de3-ad2d-6e2d8d2a9042
- Concept: a71e6bfd-6ff5-4000-8966-7c737830fab7
- Concept: 4caf23ec-c495-497e-94c1-b35995686d61
- Concept: 1b6168cd-1336-4c19-b9dc-4afc19ec77f9
- Concept: 9de4f802-1301-4b16-87c8-7de4166280af
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfecd
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfece
- Concept: d19bbdc6-45a9-4dc5-921d-9a1cc45f33bb
- Concept: 483ba4c1-5320-4f77-b983-65dbd9d15b3f
- Concept: 09f52eb4-ae51-4a2d-b34d-76f2224bd455
- Concept: 69bff2ff-fb43-4bfa-ad29-ca6c5f88501f
- Concept: c86e1175-19b2-4559-957f-bafcc4a41efc
- Concept: 5edda7d5-82c9-441f-9f5c-b18d79d91826

ConceptRoot 'Concept Name existence' (uuid=f210fea4-f81c-4333-81b9-6912585703fe)

The following entities are passing:
- Concept: e5eaa92e-d530-414f-bf86-c50a3b84fbe2
- Concept: 68ecc19c-cd8c-4a38-b0d2-8bf16d2b6455
- Concept: cb576fee-b25f-4d51-aa07-cd436851317d
- Concept: cb576fee-b25f-4d51-aa07-cd436851317c
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfecc
- Concept: b45128ae-dc95-4de3-ad2d-6e2d8d2a9042

```
- Concept: a71e6bfd-6ff5-4000-8966-7c737830fab7
- Concept: 4caf23ec-c495-497e-94c1-b35995686d61
- Concept: 1b6168cd-1336-4c19-b9dc-4afc19ec77f9
- Concept: 9de4f802-1301-4b16-87c8-7de4166280af
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfecd
- Concept: 3bb3ea16-af65-4806-9f42-4c53464bfece
- Concept: d19bbdc6-45a9-4dc5-921d-9a1cc45f33bb
- Concept: 483ba4c1-5320-4f77-b983-65dbd9d15b3f
- Concept: 09f52eb4-ae51-4a2d-b34d-76f2224bd455
- Concept: 69bff2ff-fb43-4bfa-ad29-ca6c5f88501f
- Concept: c86e1175-19b2-4559-957f-bafcc4a41efc
- Concept: 5edda7d5-82c9-441f-9f5c-b18d79d91826

## mvdXML

### ConceptRoot 'Existence of referred ConceptTemplates of Applicability in the
mvdXML file' (uuid=b3d73a9a-3693-462a-bed6-20658ed48122)

There were 1 applicable and 0 not applicable  mvdXML(s) in the checked file.

The following entities are passing:
- mvdXML: bdae3712-5c82-45fb-878f-b2f52713e96c

### ConceptRoot 'Existence of referred ConceptTemplates of Concept in the mvdXML
file' (uuid=087aae2f-2c43-4f1f-a572-42d0eb27ce2d)

The following entities are passing:
- mvdXML: bdae3712-5c82-45fb-878f-b2f52713e96c

### ConceptRoot 'Constraints ConceptRoot for Applicability Parameters'
(uuid=12e23e60-73af-48c3-ba5b-889fc9ea88b2)

There were 0 applicable and 1 not applicable  mvdXML(s) in the checked file.

### ConceptRoot 'Constraints ConceptRoot for Concept Parameters' (uuid=12e23e60-
73af-48c3-ba5b-889fc9ea88b1)

There were 0 applicable and 1 not applicable  mvdXML(s) in the checked file.

### ConceptRoot 'mvdXML Name existence' (uuid=fae1d217-26da-4cb3-aa2a-
dcc82b059c0b)

The following entities are passing:
- mvdXML: bdae3712-5c82-45fb-878f-b2f52713e96c

## ModelView

### ConceptRoot 'ModelView Name existence' (uuid=c268d637-1674-4948-8eb3-
0e185a8536a3)

The following entities are passing:
- ModelView: aa413190-fee1-4ca5-8222-bfc10369a776
```

Appendix D

The enclosed ZIP file contains the following content:

- The written part of the work as a Word document
- The created Furniture Example (XSD, Dataset XML, correct and incorrect mvdXML files)
- The created mvdXML checks (the individual files + the merged file)
- The reports of all the tests
- The source code of the developed furniture example, mvdXML checks, and checker interfaces

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Master-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 2. November 2022 Rawan Gaafar

Vorname Nachname

Rawan Gaafar

Rawan.gaafar@tum.de