Technical University of Munich

Department of Civil, Geo and Environmental Engineering

Chair of Computational Modeling and Simulation

Assuring building information quality for building analytics by translating use cases of BIM@SRE standard into the MVD format

Master's thesis

of the Master of Science program Civil Engineering

in cooperation with Siemens Real Estate

Author: Gergana Popgavrilova

Student number: [redacted]

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

2. Supervisor: M.Sc. Alexander Braun

Date of issue: 01. July 2019

Date of submission: 07. January 2020

Abstract

Evaluating large models has proven that data quality is essential for coherent analysis. This can be achieved by using open format specifications with clear definitions of terms, concepts and information requirements. The target of this master's thesis is to translate written guidelines into machine-readable exchange requirements and to investigate different methodologies to assure data quality for further automatized data evaluation. In this work, the internal Siemens Real Estate BIM guidelines and principles are analyzed and the exchange requirements for two use cases have been outlined. The extracted requirements have been firstly defined in an Excel table and mapped to IFC. Next, the exchange information has been transferred into the BIMQ platform, where software templates for properties and configuration files for Solibri ruleset SOL/203 are exported and prepared for further check of information completeness. To compare different workflows a Model View Definition (MVD) has been configured in mvdXML1.1 format, using BIMQ and in mvdXML1.2 format, using IfcDoc. Additionally, templates for other concepts such as Classification, Material Association, and Element Decomposition have been configured for the specific use cases' demand. A case study model has been validated against the generated MVD.

Zusammenfassung

Datenqualität ist essenziell für eine kohärente Analyse, wenn es sich um die Auswertung größerer Modelle handelt. Dies kann durch die Verwendung von offenen Formatspezifikationen, mit klaren Definitionen von Begriffen, Konzepten und Informationsanforderungen, erreicht werden. Das Ziel dieser Masterarbeit ist schriftliche Richtlinien in maschinenlesbare Austauschanforderungen zu übersetzen und verschiedene Methoden zu untersuchen, um die Datenqualität für eine nachfolgende automatisierte Datenauswertung zu sichern. In dieser Arbeit werden die internen Siemens Real Estate BIM Richtlinien und Prinzipien analysiert und die Austauschanforderungen für zwei Anwendungsfälle entwickelt. Die extrahierte Anforderungen wurden zunächst in einer Excel-Tabelle definiert und auf IFC Format abgebildet. Anschließend wurden Austauschinformationen in die BIMQ-Plattform übertragen, wo Softwarevorlagen für Eigenschaften und Konfigurationsdateien für den Solibri-Regelsatz SOL/203 zur weiteren Überprüfung der Vollständigkeit exportiert werden. Um verschiedene Workflows zu vergleichen, wurde eine Model View Definition (MVD) im mvdXML1.1 Format in BIMQ und im mvdXML1.2 Format mit Hilfe von IfcDoc konfiguriert. Zusätzlich wurden Vorlagen für andere Konzepte wie Klassifikation, Materialzuordnung und Element Zerlegung für die spezifischen Anwendungsfälle konfiguriert. Ein Beispielmodell wurde gegen die generierte MVD validiert.

List of Content

List of Figures

List of Tables

List of Listings

List of Abbreviations

BCF	BIM Collaboration Format
BIM	Building Information Modeling
BIM@SRE	BIM at Siemens Real Estate Guideline
bSI	buildingSMART International
CAFM	Computer Aided Facility Management
EIR	Exchange information requirements
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
IAI	International Alliance for Interoperability
IDM	Information Delivery Manual
IFC	Industry Foundation Classes
LoD	Level of Development
LoG	Level of Geometry
LoI	Level of Information
MVD	Model View Definition
SMC	Solibri Model Checker
XML	Extensible Markup Language
XSD	XML Schema Definition

1 Introduction

1.1 Motivation

The volume of the global construction output has reached a new peak in 2018 and is expected to grow by 3-4% p.a., going up to 15.5 trillion worldwide in 2030 (Global Construction Perspectives and Oxford Economics 2015). Yet, the construction sector remains the second-lowest industry on the McKinsey Global Institute industry digitalization index, with a total growth of averaged 1% over the past 20 years, meaning that the construction market needs significant innovation (McKinsey Productivity Sciences Center, Singapore 2016). However, the construction industry in Germany is facing a challenge adopting the latest technological developments, such as Building Information Modeling, despite its clear potential benefits (Bundesministerium fur Verkehr und digitale Infrastruktur 2018). As stated in a study by Roland Berger, digital technologies, such as BIM, will have the strongest impact on business models and have the potential to overcome the digitalization gap and increase the buildings' efficiency and quality (Roland Berger 2017).

Verifying buildings' design quality and accuracy against guidelines and standards has been of interest to governments and building owners, thus leading to great progress in the area of code compliance checking. Although efforts on automating code compliance have been continuing for more than 50 years, computerized methods for standard compliance checking are yet not ubiquitous and conventional in the design process (Robert Amor 11/15/2019). Nevertheless, construction guidelines and standard documentation are continuously being updated and vary from country to country. Although the amount and quality of data required are high, the power of representation and how data is structured is yet insufficiently developed. Acknowledging the broad spectrum of benefits, that machine-readable codes come with – consistency in the set of terms and clear definitions, as well as automated identification of possible deviations and anomalies, has led to further improvement in the topic of compliance checking.

Every year, Siemens Real Estate invests high amounts in the construction and expansion of office and manufacturing locations worldwide. Along the design process, however, building owners are missing transparency on planner's decisions, as well as knowledge about design alternatives and their impact on construction and operational

costs. In addition, controlling model quality and comparing it to design standards, without proper data specifications, is an iterative, time-consuming process with repetitive manual intervention on the models (Figure 1-1).

Figure 1-1 Existing analytics workflow in Siemens Real Estate (Regimantas Ramanauskas 2018)

Evaluating large models has proven that data quality is essential for a cohesive and comparable analysis. This can be achieved by using open specifications with clear definitions of terms, concepts and information requirements. The motivation behind this master's thesis is the need for an IT-based methodology, that will allow stakeholders, such as Siemens Real Estate, to carry out automated systematic value engineering and assure data quality for building models at early project phases. To address these issues, this work aims to explore the possibilities that modern BIM tools give to establish an automated methodology for generation, capturing, processing and evaluating of data.

1.2 Scope

The scope of this work is to develop, implement and evaluate methods for increased automation of digital data validation and thus assure information quality by translating quantitative and qualitative data from BIM@SRE. Definitions of object types, such as façade, windows, walls, doors and spaces, will be extracted from the BIM@SRE and shall be outlined utilizing the overall IFC schema and further defined in a Model View Definition (MVD) (Figure 1-2). Two approaches are thoroughly studied and compared in terms of flexibility and applicability.

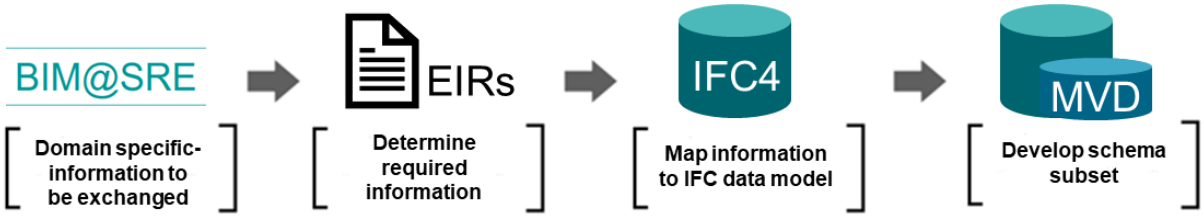

Figure 1-2 Pursued analytics workflow in Siemens Real Estate

The focus lies on defining use cases from written BIM guidelines and transforming them into digital form, thus allowing companies to assure data quality and compare design and cost alternatives during project progress. The output of this work, in form of documents and future standardization proposals, contains a general definition of Siemens employer's information requirements, generation of rulesets for property checks, specification and validation of exchange requirements in mvdXML format.

1.3 Structure of work

The second chapter (Chapter 2) offers an overview of the internal Siemens Real Estate BIM guidelines. The current workflow and practices have been examined and the reasons for insufficient model quality have been outlined. Two BIM use cases for data exchange have been defined and elaborated.

In Chapter 3 the technical background of the open exchange format Industry Foundation Classes (IFC) and the IFC Shema is introduced, as well as the correlation between IFC and the Model View Definitions (MVD). Additionally, the methodology of the IDM and EIR's, and their crucial role in the implementation of vendor-neutral processes, is clarified. In conclusion, the importance of standardization through the IFC format for the interoperability in the construction industry is elaborated.

The groundwork of this master's thesis is explained in Chapter 4, where the motivation and choice of the conceptual approaches are explained. Exchange requirements for defined use cases have been defined in an Excel table and mapped to IFC. Next, the information has been transferred into the BIMQ platform, where software templates for property import and Solibri ruleset's configuration files are exported for further check of information completeness. Templates for other concepts such as Classification, Material and Element Decomposition have been configured for the specific use cases' demand. To compare different workflows a Model View Definition (MVD) has been configured in mvdXML1.1 format in BIMQ and mvdXML1.2 format, using IfcDoc.

Chapter 5 shows the implementation and validation methods on a case study model, from which and IFC has been exported and validated against Solibri checking rules, mvdXML1.1 and mvdXML1.2. In conclusion, the limitations, dissimilarities und suggestions for future work are discussed in Chapter 6.

2 Introduction to Industry Foundation Classes, Information Delivery Manual and Model View Definition

2.1 Background

Efforts to improve data exchange between project participants date back to 1996s when a group of stakeholders started examining the potential of making different applications work together (Autodesk Inc). Definitions of file formats for instances and uniform information processing interfaces, together with the development of a standard for the Exchange of Product model data (STEP), have been documented in the ISO 10303 standard (Borrmann et al. 2018, p. 85). The earliest developed formats were limited to exchanging primarily geometric data, but the rapidly developing industry at that time demanded a more dynamic loss-free data exchange. As a result, the International Alliance for Interoperability (IAI), now known as buildingSMART International (bSI), has been founded, to support information exchange based on non-proprietary formats. The first version has been issued as the Industry Foundation Classes (IFC) – a standardized, comprehensive data format for exchanging digital data in a vendor-neutral way, thus making it an essential part of the Building Information Modeling methodology (Borrmann et al. 2018, p. 12). Since its establishment as a vendor-neutral standard, independent from ISO standardizations and available free of cost, IFC has been implemented for import and export in numerous vendors' products, with the most common supported version IFC2x3, currently being replaced by IFC4 – ISO Standard since 2013.

2.2 Industry Foundation Classes

Industry Foundation Classes (IFC) is a vendor-neutral format and a global standard designed to describe, share and exchange design, construction and facility management information (buildingSMART International 2019i). Based on the IFC4 standard an international norm DIN EN ISO 16739 has been developed, containing terms, concepts and data descriptive elements, derived from practical experience in different fields of the building industry and facility management. The standard also defines a set of data schemas, used to digitally specify building data, based on the EXPRESS language (DIN EN ISO 16739).

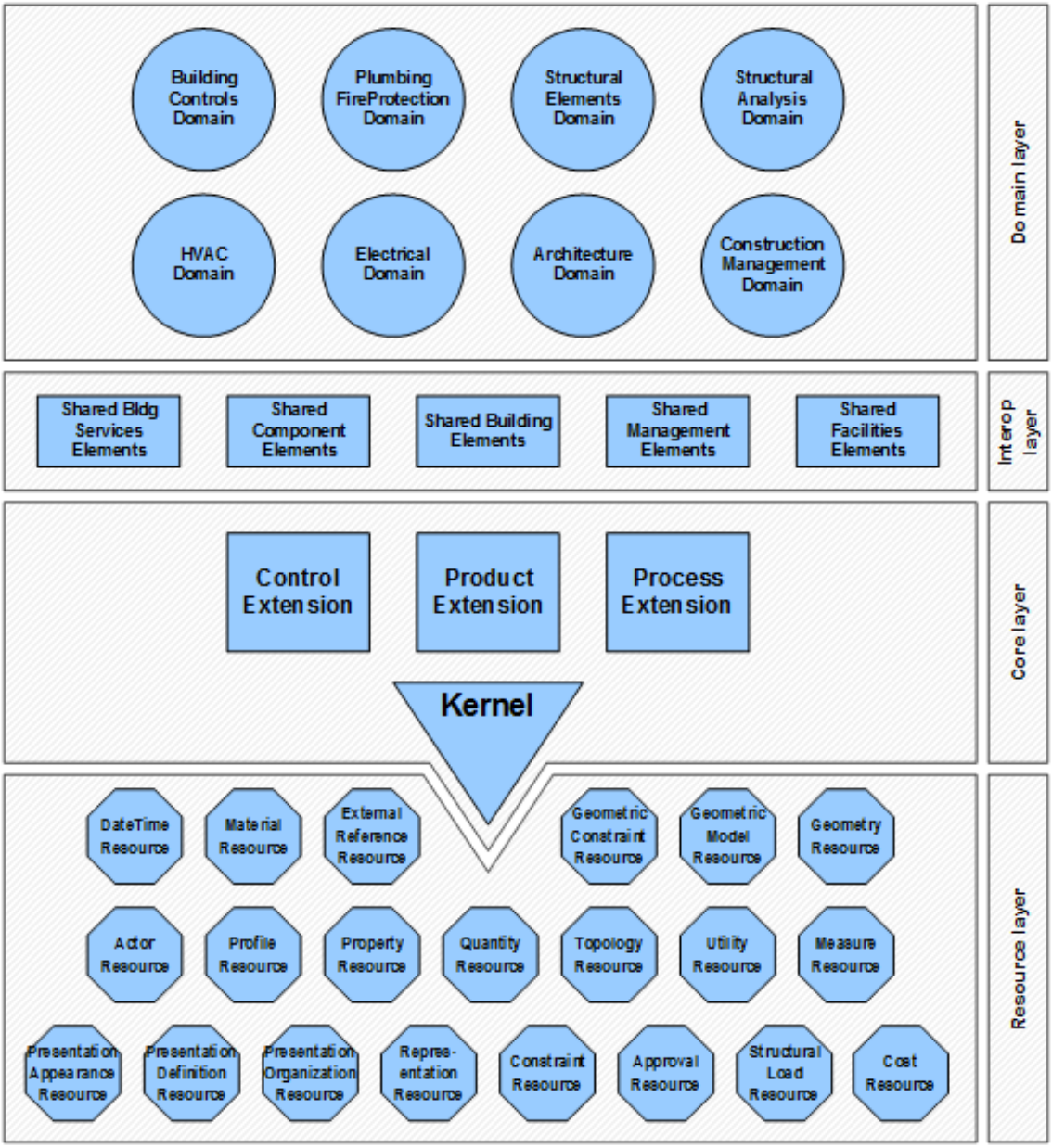

Figure 2-1 Data schema architecture with conceptual layers (buildingSMART International 2019e)

The IFC schema contains objects and property definitions, and relationships between different entities. The structure of the IFC schema is both complex and extensible, meaning it can be continuously added to, without affecting the existing core structure. The data model is organized in layers, that containing classes, that can be referenced by the layers above (Figure 2-1) (Borrmann et al. 2018, p. 90). The IFC4 Domain layer consists of highly specialized classes, that only apply to particular domains such as architecture, building control, HVAC systems and structural elements. The Interoperability Layer represents an interoperability layer between the core of the data schema (The Core Layer) and the domain-specific schemas (Domain Layer). Its classes derive

from the Core Layer and can be used by different application schemes. The Core layer contains classes, defining the basic structure, the key relationships and general concepts, that can be further specified and reused. The Kernel schema consists of abstracts classes, such as *IfcRoot, IfcObject, IfcActor, IfcRelationship,* etc., that also build the three scheme extensions Product Extension, Process Extension and Control Extension. The lowest level is taken by the Resource Layer, which contains schemas, that can be referenced by all other layers. Its classes cannot exist independently, since they do not derive from *IfcRoot*, but can be referenced by objects that do derive from a subclass of IfcRoot. Examples include Material Resource, Geometry resource, Topology Resource and others.

2.3 Information Delivery Manual

The Information Delivery Manual is another standard created by buildingSMART, that proposes a methodology on how information is created and shared by participants. Documented in the ISO 29481-1:2010 “Building information modeling - Information delivery manual - Part 1: Methodology and format” (buildingSMART International, BLIS Consortium/Richard See 2012; DIN ISO 29481-1). IDM is specifically designed as a methodology to meet the needs of all involved in a building project and to support the process of information exchange. For example, planers or general contractors, need to know which IFC components are essential for their needs, while solution providers – software vendors, shall implement those features to meet the needs of the users.

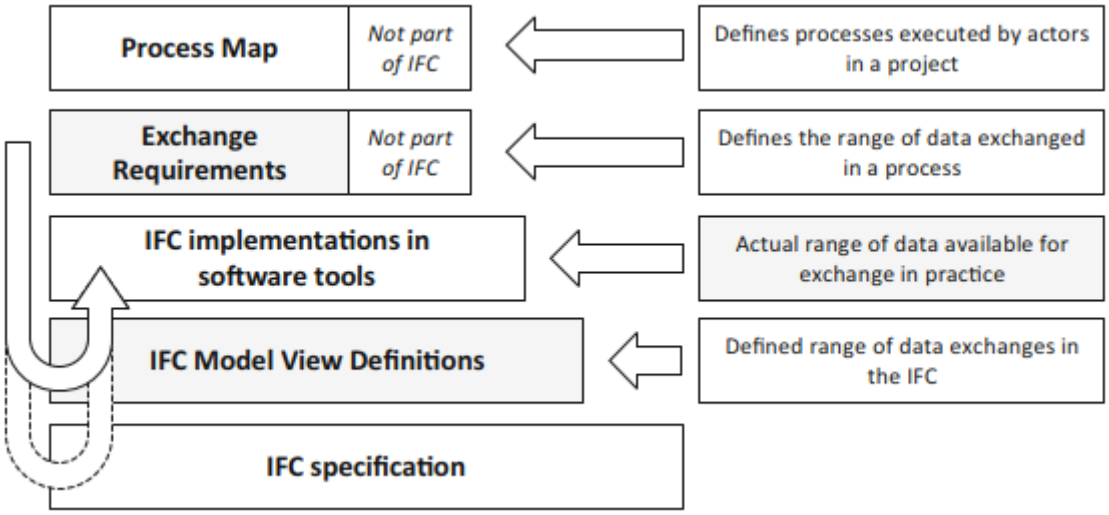

Figure 2-2 Correlation between elements of the IDM methodology (Borrmann et al. 2018, p. 131)

IDM has been created to resolve collaboration issues in construction projects, by capturing business processes and providing user-defined specifications on which data, at what point and by whom is to be delivered. These exchange requirements also indicate what data part of the sub-schema is needed to fulfill such an exchange (Chipman et al. 2016). A Process Map describes the relationship between these requirements and the business project process. Once expert's knowledge and experience are collected and structured in a human-readable document, they are converted into a machine-readable technical solution, based on consistent and reusable sets of IFC concepts. To import and export these exchange requirements, software providers require additional technical information in terms of IFC capabilities. The correlation between the single components of the IDM methodology is shown in Figure 2-2.

2.4 Model View Definition

A Model View Definition defines a subset, or a "filtered" view, of the overall IFC schema, and contains implementation agreements for objects, properties and relationships, needed to depict exchange requirements in a BIM use case. MVDs vary in their scale, depending on the particular purpose. For example, archiving a project can involve the entire schema, whereas establishing micro workflows (cost evaluation or space efficiency) between different actors in large companies, can be limited to outlining and defining a few object types and associated data. The wide spectrum of the schema allows for the user to represent construction and building information in diverse ways and details (buildingSMART International). MVDs are intended to narrow the scope, deepening on the use case or the participant and allow a consistent and aligned with the IFC schema data exchange. The defined MVD contains objects, representations, relationships and attributes and responsible actors, and can be referenced as contract deliverables. However, the MVDs can only filter and describe the required data that is to be included and cannot guarantee its correctness and consistency. The data exported by applications can be validated against the requirements of an MVD by generating an mvdXML.

MVDs can be used by software vendors to implement an IFC export and import support to align with predefined exchange requirements and utilize the need of different stakeholders to determine the scope and format of the data they need to exchange. Since not all participants need the same information delivered and received, software tools were intentionally developed to serve diverse functions. Considering that not every

software has import and export functions, that support all entities in each domain, buildingSMART has created different MVDs that define what parts of the IFC schema are implemented for diverse purposes.

BuildingSMART International has developed official Model View Definitions, based on the IFC2x3 and IFC4 schema. The IFC4 Reference View 1.2 was created for the design coordination between architectural, structural and building services (MEP) domains. The RV subset of the IFC schema suits workflows, where requests on data exchange information are sent back to the source and handled by the authentic author. Such workflows may include Coordination planning, Clash detection, Quantity take-off and others. The RV enables information definitions with rich content of property information and shape representations. Semantic model elements can be further specialized using its *PredefinedType* enumeration or a user-defined type. Additionally, the concept *Object Typing* associates element occurrences with a correspondent element type and allows for type occurrences to describe and share common model information like geometric shape, property and material information. Other concept templates are Software Identity, Project Units, Property sets and Quantity sets, Material association, Element decomposition and other geometry related concepts.

The IFC 4 Design Transfer View includes more advanced geometric representations to enable the transfer of information from one tool to another (buildingSMART International 2019k). The overall goal is to assure a workflow in which models are transferred and further modified by the same software platforms and tools. The DTV focuses on capturing and exchanging rich geometry preserving parameters for major building elements, together with design specifications for all building components.

2.4.1 Mvdxml

An mvdXML is a standardized format to define and exchange MVDs with exchange requirements and validation rules (Chipman et al. 2016). The mvdXML format is a structure, applied to the IFC data schema and represents an MVD and its associated ERs. In general, an mvdXML format captures exchange definitions for import and export scenarios and can enable automated validation of IFC models against quality assurance and software certification requirements (Figure 2-3).

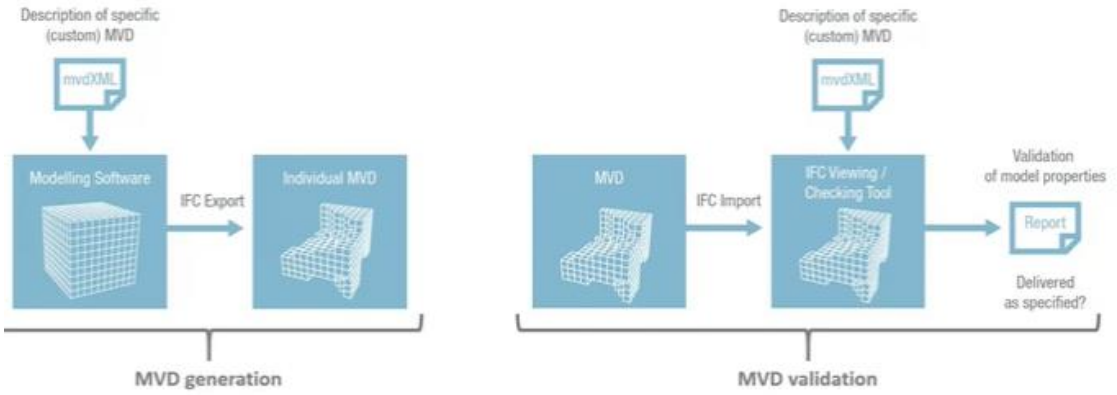

Figure 2-3 MVD generation and validation (Baldwin 2017)

According to the user needs, data can be filtered to reduce information and provide only required data for a specific software application (Baumgärtel et al. 2016). The main objective, that nowadays mvdXML is used for, is to automate the process of defining machine-readable information requirements and replace error-prone manual efforts, time-consuming and long-term unrealizable, due to the dynamical character of specifications.

The mvdXML format is based on the definition of the following elements:

- *ConceptRoots* - outline the entities, containing information like attributes, geometric shapes, dynamic property sets and other semantics, of the IFC schema incorporated in the model view.
- *ConceptTemplate* - a reusable list, that captures concepts, applicable across multiple *ConceptRoots* and consists of attributes and other entity definitions, required to build a functional unit.
- *Concept* - refers to a *ConceptTemplate* and describes template rules for common subsets of information and specifies the particular constraint and usage for each entity.
- *ExchangeRequirements* - describe how concepts are to be handled for each entity and each exchange.

The structure of an mvdXML file starts with an instance of mvdXML that defines two main sub-elements: *Templates and Views. Templates* is *a* list of reusable *ConceptTemplates,* whereas *Views* represents a list of model view definitions, called *ModelView*, that contains only the specific entities and associated concepts, that define a particular exchange requirement (Chipman et al. 2016). Hereafter a simplified overview of the main components of an mvdXML is presented *(*Figure 2-4*).*

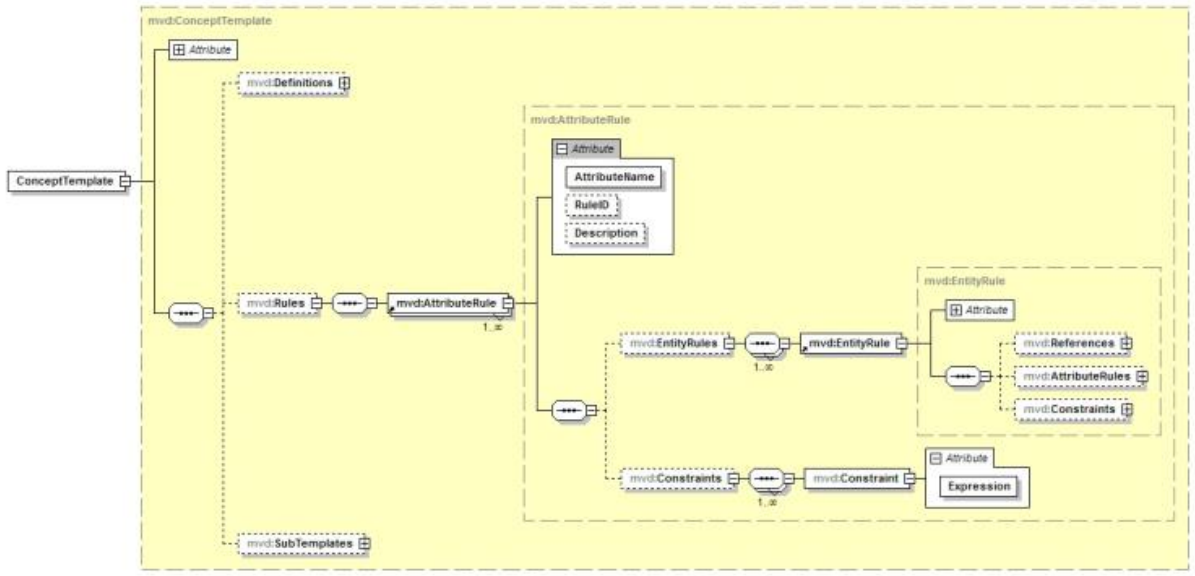

Figure 2-4 mvdXML elements for ConceptTemplate (Chipman et al. 2016)

*ConceptTemplate*s contain elements such as:

  - uuid: a universally unique identifier, used to reference the *ConceptTemplate*.
  - applicableSchema, such as IFC2x3 or IFC4.
  - applicableEntity: root entity of the *ConceptTemplate*, deriving from *IfcRoot*.
- Definition: a human-readable description of the use of the ConceptTemplate.

```
    <ConceptTemplate uuid="ccb45aa2-74e1-46e8-8e35-ca5e5d7cc9dd" name="Ob-
ject Attributes" status="sample" applicableSchema="IFC4" applica-
bleEntity="IfcRoot">
      <Definitions>
        <Definition>
          <Body><![CDATA[<p> All entities having semantic significance de-
rive from <i>IfcRoot</i>, where instances are identifiable within a data
set using a compressed globally unique identifier (IFC-GUID). This identi-
fier must never change during the lifetime of an object, which allows data
to be merged, versioned, or referenced from other locations.</p>]]></Body>
        </Definition>
      </Definitions>
```

Listing 2-1 ConceptTemplate elements in mvdXML, Example 1

- Sub Templates: an optional concept template, that extends the definition of the main *ConceptTemplate*, allowing to group multiple *ConceptTemplates* under a common criterion.

```
      <SubTemplates>
        <ConceptTemplate uuid="c19ec186-9cfd-47fc-a4d4-9fb35008d04a"
name="Object User Identity" status="sample" applicableSchema="IFC4" appli-
cableEntity="IfcObject">
          <Definitions>
            <Definition>
              <Body><![CDATA[<p>]]></Body>
            </Definition>
          </Definitions>
```

Listing 2-2 ConceptTemplate elements in mvdXML, Example 2

- *Rules*: a list of definitions of attributes or relationships of the root entity, building a tree structure, that consists of *AttributeRules*, referring to *EntityRules*, referring to *AttributeRules*, and so on. Here of significant importance are the:
  - *AttributeName:* the name of the attribute or the relationship.
  - *RuleID:* allows to document a specific usage of the entity or validate its value against ER's in the *ModelView* section.
  - *Constraints*: a set of constraints on the schema population, if used.
  - *EntityName:* the name of the underlying type.

```
<Rules>
  <AttributeRule RuleID="ObjectName" AttributeName="Name">
    <EntityRules>
       <EntityRule EntityName="IfcLabel" />
    </EntityRules>
   </AttributeRule>
 <AttributeRule RuleID="ObjectDescription" AttributeName="Description">
     <EntityRules>
     <EntityRule EntityName="IfcText" />
    </EntityRules>
  </AttributeRule>
```

Listing 2-3 ConceptTemplate elements in mvdXML, Example 3

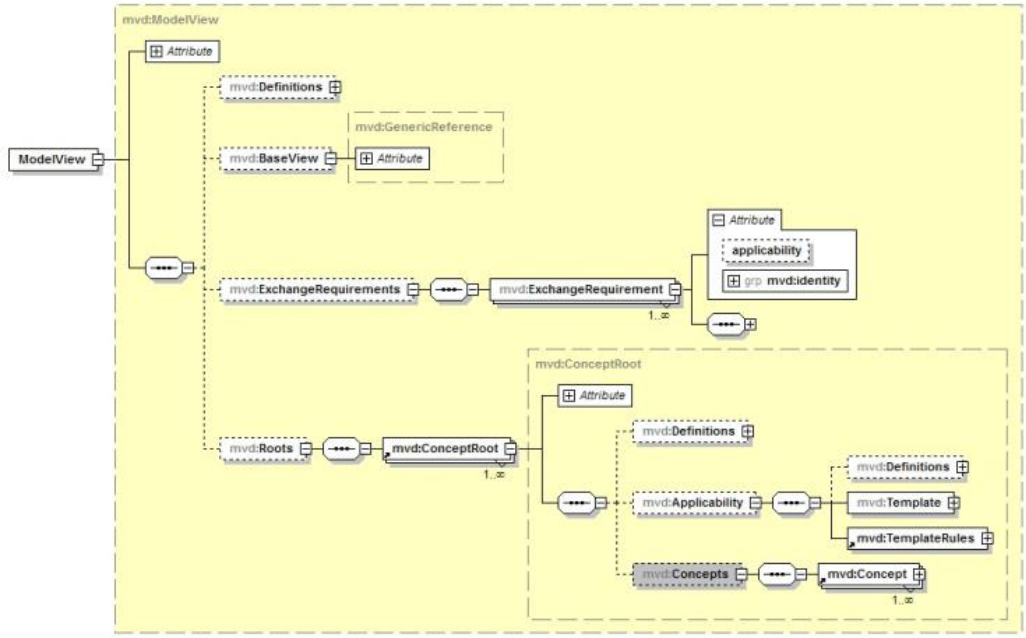

Figure 2-5 mvdXML elements for ModelView (Chipman et al. 2016)

The *ModelView* element describes how concept templates are to be used in a particular Model View Definition (Figure 2-5). Some of its elements are:

  - uuid: a universally unique identifier of the Model View.
  - applicableSchema: such as IFC2x3 or IFC4.
- *Definition*: a human-readable description of the purpose of generating the MVD documentation.
- *BaseView: a* reference to another model view definition, that indicates if it is an add-on view with defined restrictions or an extension of the model view.
- *ExchangeRequirements*: a list of exchange requirements, defining if and how concept' specifications are to be fulfilled for each ER.
- *Roots*: a list of *ConceptRoot* entities, defining concepts and their template rules, applicable to each IFC.

```
    <ModelView uuid="1eb5ae5b-d0c9-4a15-965d-d8a2110614ff" name="Siemens
SRE MVD" status="sample" applicableSchema="IFC4">
        <Definitions>
        <Definition>
          <Body><![CDATA[Specification of exchange requirements for spaces,
walls, facades, plates, mullions, windows and doors)]]></Body>
        </Definition>
      </Definitions>
      <BaseView>86aa67de-829b-467b-9cd8-dd364beabe79</BaseView>
      <ExchangeRequirements>
        <ExchangeRequirement uuid="42008c61-7b8f-4c3d-8aa2-7b99409dfba1"
name="BIM2Planon" status="sample" applicability="export" />
        <ExchangeRequirement uuid="191d5e3b-08e7-47cf-afd6-d2ec9a62df58"
name="Building Model Analyzer" status="sample" applicability="export" />
      </ExchangeRequirements>
      <Roots>
```

Listing 2-4 ModelView elements in mvdXML, Example 1

- A *ConceptRoot* references to a specific IFC entity, e.g. *IfcSpace*, as a model element in an MVD and contains:
  - *Applicability*: a list of *TemplateRules*, linked to an applicable *ConceptTemplate*. Here additional constraints can be defined, that need to be fulfilled before concepts are validated. Listing 2-5 shows a defined constraint on its *PredefinedType ='.EXTERNAL.'*. how the element Courtyard, mapped to the entity IfcSpace, is differentiated by setting

```
        <ConceptRoot uuid="560634f9-63f1-4c1e-a6eb-fe80f22f778d"
name="Courtyard" status="sample" applicableRootEntity="IfcSpace">
          <Applicability uuid="00000000-0000-0000-0000-000000000001"
status="sample">
            <Template ref="00000000-0000-0000-0001-000000000001" />
            <TemplateRules operator="and">
              <TemplateRule Parameters="PredefinedType[Value]='EXTERNAL'
AND T_PredefinedType[Value]='EXTERNAL'" />
            </TemplateRules>
          </Applicability>
```

Listing 2-5 ModelView elements in mvdXML, Example 2

  - *Concepts*: a set of concepts, which describe template rules for subsets of information, within the context of the particular concept root (Chipman et al. 2016).
    - *Template*: provides a reference to a *ConceptTemplate*, where the parameters, further defined as part of this concept, are contained.
    - *Requirements*: define how a concept is to be handled for each exchange

```
          <Concepts>
            <Concept uuid="93f242f3-2daf-4d4b-9dc7-f8c0ecc743c8"
name="Classification" status="sample" override="false">
              <Template ref="4a224609-6578-4c75-afcf-8affa86e5ef2" />
              <Requirements>
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="7fab58cb-4655-489a-862d-85e26a3096b2" />
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="cfa72802-9d70-407c-9616-56f5b53f8708" />
              </Requirements>
```

Listing 2-6 ModelView elements in mvdXML, Example 3

- *TemplateRules*: a tree of *TemplateRule*, following a Boolean logic between individual template rules. The outermost *TemplateRules* element hast to be validated as true, to pass the validation. Listing 2-7 shows defined constraints for value existence and value syntax for different parameters.

```
              <TemplateRules operator="and">
                <TemplateRule xsi:type="TemplateItem"
Parameters="ClassificationName[Value]='CAFM'">
                  <References />
                </TemplateRule>
                <TemplateRule xsi:type="TemplateItem"
Parameters="ClassificationName[Value]='Uniclass'">
                  <References />
                </TemplateRule>
                <TemplateRule xsi:type="TemplateItem"
Parameters="Name[Exists]=TRUE">
                  <References />
                </TemplateRule>
                <TemplateRule xsi:type="TemplateItem"
Parameters="Identification[Exists]=TRUE">
                  <References />
                </TemplateRule>
              </TemplateRules>
```

Listing 2-7 ModelView elements in mvdXML, Example 4

The rule-based syntax of the mvdXML format allows further specification of *TemplateRule.Parameters* and *Constraint.Expression*. The parser rules describe metric values and operators and how they apply to different data types:

- Value: specifies the value of the attribute.
- Size: specifies the size of a collection.
- Type: specifies the type of the value assigned to the attribute.
- Unique: indicates whether a value is unique within the population of instances in the XML file
- Exists: indicates if an attribute or an entity exists in the XML file

The main scope of the first published release of mvdXML1.0 in 2013 has been to define concepts and concept tables, include entities and attributes, generate MVD

documentation and support the development of MVD specific IFC subset schemas. The focus of the next update - mvdXML1.1, has shifted from exclusively generating MVD documentation to creating mvdXML files with the purpose of IFC data validation against MVDs (buildingSMART International 2019l). A grammar for configuration rules has been introduced, including simplified parsing of complex expressions and descriptions for metric values (Chipman et al. 2016). The free combination of logical operators, the implementation of partial templates and further filter criteria, has allowed more flexibility and readability (Matthias Weise 2019).

Mvdxml files can be created by using text or XML editors. However, manually defining the parts of a subschema requires deep knowledge of the IFC schema and XSD format structure. BuildingSMART provides an application, IfcDoc, to generate mvdXML based on simplified ifcXML editing.

2.5 BuildingSMART Data dictionary (bsDD) and BIM Collaboration Format (BCF)

In addition to the IFC schema and the Model View Definition methodology, bSI is also developing other data standards to support data exchange in construction project processes, based on open formats. To ease the information management bSI created a database, BuildingSMART Data dictionary (bsDD), that incorporates standardized property set definitions, available in different languages. bsDD is used for extending the IFC model, by directly referencing properties and product information in external data structures (buildingSMART International 2019a). The process of defining classification systems on an international level and linking its terms to bsDD is based on the IFD standard (ISO 12006 part III) (buildingSMART International 2019b). An advantage of such a central repository for property sets is the possibility to directly link manufacturer information, building codes, etc. to model elements. bsDD provides several regional or use-case specific classifications, properties and property sets. However, additional user-specific properties need to be defined separately and made part of the ERs. Organizing this amount of diverse specifications in a uniform and consistent structure and providing an automated method of handling it, is a challenge that the field of Semantic Web and Linked data is currently working on.

The BIM Collaboration Format (BCF) was developed as a data format and web-based service, to communicate project issues, such as element collisions, or modeling problems, between actors. Although the standard support of BCF is not yet officially certified

(Borrmann et al. 2018), many commercial software tools have already implemented BCF in their workflow.

2.6 Literature review and recent work

2.6.1 MVD development

Depending on the use and the purpose of creation, MVDs can vary in the methodology of creation and comprehensiveness. In following the recent work of MVD developers is summarised and valuable conclusions for the further scope of this work are outlined.

Solihin et al. (2015) developed a methodology to estimate data quality of IFC data in terms of conformance with a set of agreed standards. The research also includes suggestions for rules to estimate data completeness and correctness. According to Lee et al. (2016), there has been a need for a standardized method to link exchange requirements to information elements in MVD, therefore a new approach of formalizing domain knowledge and a method to include ontologies to create an MVD was suggested.

Zhang et al. (2013) present an overview of MVD generation methods. Next to the mvdXML format, the functionalities and differences of methods such as xPPM (eXtended Process to Product Modeling), GMSD (Generalized Model Subset Definition) and Semantic Exchange Module (SEM) were elaborated. Zhang and Beetz (2014) implemented a mvdXMLChecker prototype to parse MVD based on mvdXML and check IFC models. It has been stated that most of the existing model view concepts can limit the scope of the IFC schema to subsets, but since similar rules can be represented in multiple ways, the mvdXML structure is not strict enough to validate models. In conclusion, it has been stated that logic theory-based methods, such as ontology and semantic web technologies, can be used to formalize the model view concepts (Zhang et al. 2015). Jiang et al. (2019) researched the code checking possibilities for green constructions. Based on the level of difficulty to meet the requirements of green construction clauses, they can be classified into four types. Approaches based on mvdXML or semantic technology were adopted for the appropriate inspection of each type. Furthermore, validation of models can be extended by reducing the information that is from importance and creating a new partial building information model. This method of MVD based filtering, presented by Baumgärtel et al. (2016), is based on converting mvdXML files into ifcQL commands. Studies on different types of filtering data (schema level, class level or object level) have been conducted by Windisch et al. (2012).

Independently from technological approaches and researches, a number of IFC-based BIM standards have been developed on a national, company- or project-specific level (Rijksgebouwendienst 2012; STATSBYGG 2019). Based on these agreements efforts have been made to define exchange requirements, that fulfill a specific context (STATSBYGG). Praxis-related examples of rule interpretation processes and MVD development have been conducted by groups in various fields of work, such as automated code-checking procedures for conformance with local building regulations and laws (Zhang et al. 2015), bridge inspection systems (Sacks et al. 2018), environmental and energy performance assessment (Pinheiro et al. 2015) and checking fire-safety and pedestrian simulation requirements (Sina Pfuhl October 2018b; Jimmy Abualdenien et al. 2019).

2.6.2 IDM implementation

Efforts have been made to unify common use-cases and build exchange requirements templates. An initiative of the IDM Configurator Group of bSI aims to create an IDM/MVD functional tool, that allows users to directly generate IDM and MVD, based on a structural framework. The goals of the group are to develop an international standard for exchanging IDM and MVD data and to provide software developers, IDM/MVD/BEP developers and other stakeholders with a guideline on how to develop and share their outcomes. The demand-driven creation of such a tool can allow a seamless development of Process maps, exchange requirements, and MVDs from employer's information requirements, as well as the development of IDM/MVD by re-using and modifying existing IDMs/MVDs.

buildingSMART Switzerland has conducted efforts into enabling a global collection of existing use cases and the creation of new ones by the roll-out of a Use Case Management tool, developed by (Richard Kelly 2019). The tool aims to ease and speed up the process of defining project-specific use cases and their further implementation into exchange requirements. There are multiple levels in defining use cases and data exchange, which makes the environment for exchange requirements very dynamic and contrasting in different sectors and countries. An innovative approach, developed by J. Abualdenien and A. Borrmann (2019), presents a multi-LOD meta-model and introduces the concept of Building Development Level (BDL), that comprises components with diverse Levels of Development, required for a specific design stage.

The first steps have been also made in specifying minimum information requirements on a national level. The Federation of the German Construction Industry has presented a technical position paper "BIM in building construction" for the standardization of future construction processes (Die Deutsche Bauindustrie 2019). In addition to defining BIM use cases, data exchange scenarios and guidelines on model quality insurance, the document also derives recommendations for exchange information requirements and minimum semantic information requirements on model elements. Nonetheless, no correlation or mapping to open data formats have been provided, thus presenting the risk of requirements being defined differently and redundantly by each stakeholder. The growing demand to implement the IDM methodology has also been recognized by BIM locket, who together with buildingSMART Benelux has developed a publicly available BIM basic IDM in more than 14 languages (BIM Loket 2019). The document is aligned with the IFC schema and outlines minimum information requirements consisting of correct naming and suggestions on how model and capturing semantic data.

2.7 Interoperability through open standards

The foundation of data interoperability and data exchange has been formed by the Industry Foundation Classes. IFC plays a vital part in the digitalization backbone towards an integrated digital process workflow in the construction branch. The need and importance of open standards for the AEC industry have been acknowledged since the establishment of BIM as an innovative planning method in the construction industry. One of the important features of Building Information Modeling is ensuring the model's semantic richness, of information such as quantity, cost and time and assure an error-free exchange of this data between software products and stakeholders. Richard Kelly, the operations director at buildingSMART International, points out that companies are effectively analyzing only 1% of their data, the other 99%, called dark data, is collected and stored but not used in any way (Richard Kelly 2019). Since construction projects' complexity is constantly developing, sufficient information exchange between stakeholders is crucial for the interoperability in the sector. Digital ways of supporting the process of exchanging information in projects are provided in the ISO standard 19650. The standard talks about how information should be specified, delivered and controlled.

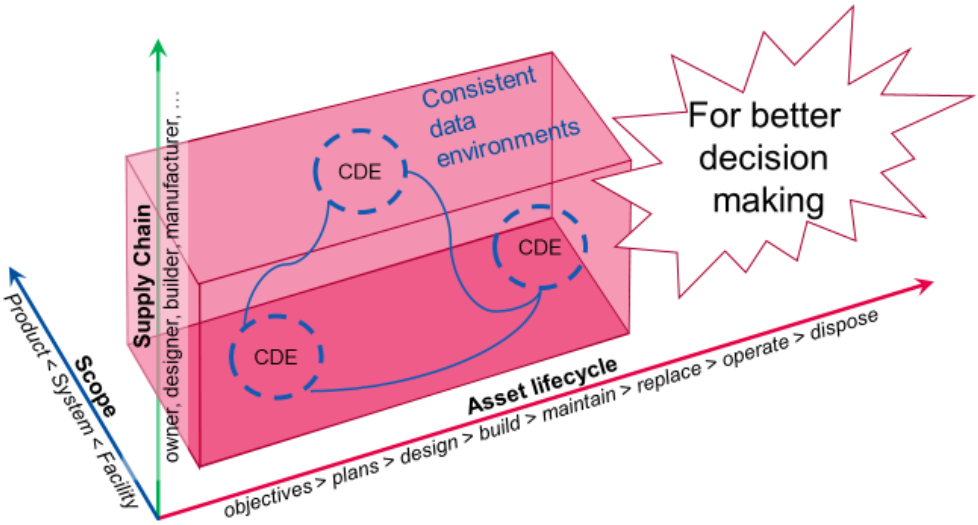

Figure 2-6 Representation of the digital environment in 3 dimensions (Richard Kelly 2019)

Open Standards have an impact also on project decision making in the field of real estate. The digital environment can be represented in 3 dimensions, as visualized in Figure 2-6. The implementation of openBIM standards aims to link the project's scope, different participants on the supply chain and asset's lifestyle through a consistent data environment and allow an infallible information exchange flow for better decision-making regarding cost, performance and risk to be taken.

3 Building Information Modeling in Siemens BIM@SRE

3.1 Scope of BIM@SRE Guideline

Siemens Real Estate has embraced the need for digitalization in the construction sector by implementing BIM methodology for the whole lifecycle of new buildings. BIM@SRE was created as a project independent guideline that defines the use of BIM in real estate projects. The guideline describes a number of technical, methodical and information-based requirements, incorporated in four chapters – General Conditions, BIM Use Cases, Modeling Standard and Data Management (Figure 3-1)

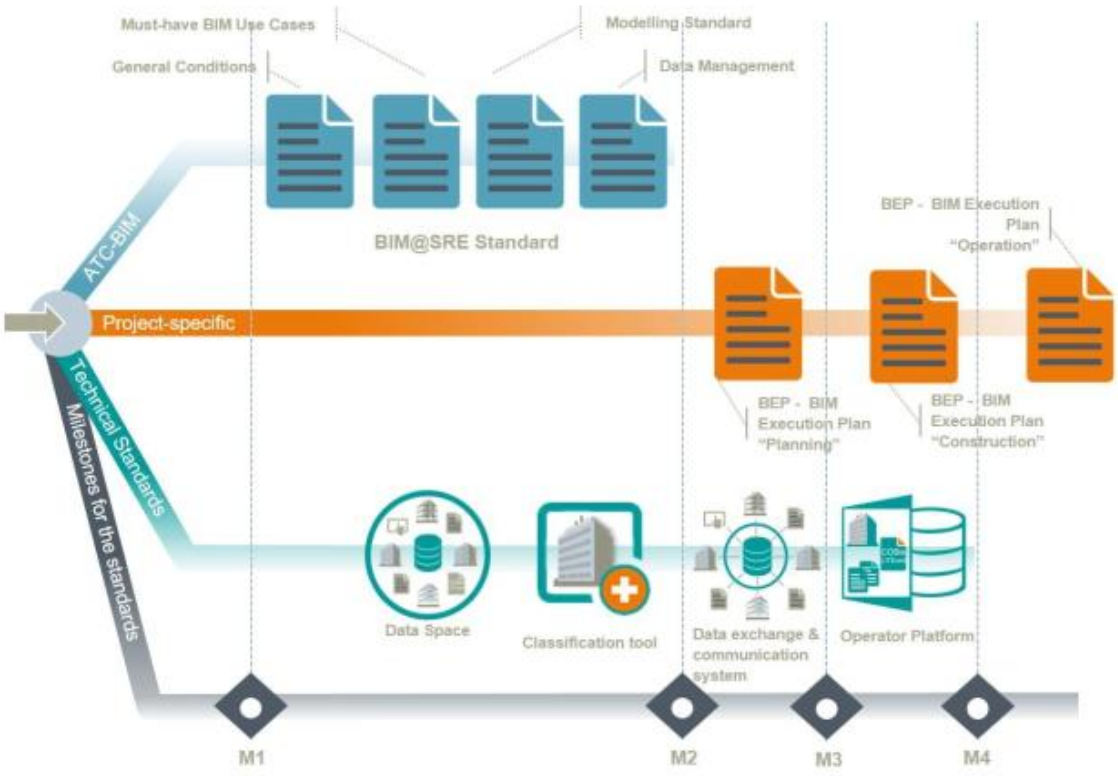

Figure 3-1 Overview of documents and technical project standards (Siemens AG Real Estate 2017)

The first chapter gives detailed information on the general BIM Process and an overall Process Map, defines data drops for each of the project phases - planning, construction and operation, and specifies responsibilities and tasks for each BIM role. Chapter A.4 refers to the BIM documents and technical project standard, that explain the requirements of Siemens Real Estate rentable space standards, as well as technical requirements which must be met, as part of a contract. Additional project documentation includes project-specific guidelines – along with the creation of tree BIM Execution

Plans in each project phase, as well as technical standards for Classification and Data exchange.

3.2 Use cases

Chapter two of the guideline lists the possible BIM Use Cases applied in SRE projects, including their implementation and contractor deliverables. Described use cases sum up to 14, including various topics such as KPI's; Hard and Soft clashes; Bill-of-quantities; Collaboration, Coordination and Communication. Must-have input for each use case includes the documentation of the Facility Service and Facility-relate building functional elements (if relevant for the use case), so as documentation of component specifications in Room and Equipment schedules (Siemens AG Real Estate 2017). The definition of a functional element is not specified, but its use in the guideline, suggests, that it includes building elements with physical representation (walls, windows, etc.), as well as more abstract concepts like spaces, floors and zones.

3.3 Modeling standard

The Modeling Guidelines contain requirements for the model structures, classification and information standards and defines the levels of detail and level of information for BIM models. For this master's thesis the architectural model definition's content in BIM@SRE has been outlined below (Figure 3-2).

Model designation		
3D models	Model content	Allocation
Site model	e.g. streets and walkways, parking spaces, vegetation, terrain	
Room model	Basic interior construction, modular interior design, roof, façade e.g. non-load-bearing walls, suspended ceilings, windows, doors, façade, stairs, rooms, permanently installed furnishings	Architecture
Equipment (furniture, machines, etc.)		
Architectural model		
Supporting structure model	Structural work, e.g. foundations, floors, pillars, load-bearing walls, breakthrough openings	Supporting structure
Heating system model	e.g. circular main, riser, stationery heating surfaces, reheater	Building services
Refrigeration system model	e.g. circular main, riser, functional element activation, sealing radiant cooling panels, air-conditioning equipment	
Room air ventilation model	Ventilation ducts and elements (e.g. fire dampers, volume flow controllers, reheaters)	
Plumbing model	e.g. wastewater/freshwater, rainwater/ greasy wastewater (piping), facilities	
Electrical installation model	e.g. horizontal and vertical cable channels, luminaires, manholes, distribution panels, transformer.	

Figure 3-2 Content information of 3D models according to BIM@SRE (Siemens AG Real Estate 2017)

3.3.1 Model building tool

To prevent quality inconsistencies all model elements must be created using element-specific software tools, for example, walls are created using a wall tool and not a slab tool. Elements that lack a specific creation tool or functional elements with complex geometries, shall be assigned to the most similar element category or using the 'General Model' category. Moreover, all modeled elements should be structured based on building levels, so that data can be extracted floor by floor. These modeling techniques are from essential importance for immaculate quantity take-off and classification.

As per market studies conducted upon the creation of the guideline, Autodesk Revit was the most commonly used application worldwide (Siemens AG Real Estate 2017, p. 4). Products from the Autodesk suite were already being used by several Siemens divisions, which has led to the decisions that currently deliverables must be submitted in .rvt format. The Closed BIM method is applied as the mandatory method for all deliverables to assure that the operator's need can be accounted for appropriately. All models must be created using the Autodesk Revit software, except for service providers (General Design and General Constructor) internal processes and coordination with additional third-party planners. Discipline-specific plans with exclusively geometric relevance for the implementation of the BIM@SRE standards may also be created using non-Autodesk Revit software. However, it must be ensured that they can be included in the model quality assurance process (Siemens AG Real Estate 2017).

3.3.2 Rooms, floors and zones

Specifications of rooms, floors and zones are of great importance for real estate projects. When modeling rooms, the following minimum requirements must be met (Siemens AG Real Estate 2017, p. 60):

- A model should not contain any open volumes or overlaps, meaning that all rooms, shaft sections and even rooms without a limiting element placed on each side, such as courtyards or terraces, must be created floor-based as 3D rooms.
- Room naming is to be assigned consistently and must conform to the function and information linked to the specified room.
- Room classification must conform to the SRE room categories. For example, circulation zones in open office areas can be classified separately and

accounted for individually in evaluations, depending on their usage (escape routes, open office common area or other traffic areas).

- Interior finishing elements for basic and modular interior completion, such as Think Tanks, individual offices, Phone boxes, etc. must likewise be defined as rooms. If there are several different types of flooring planned in a single space, space should be divided accordingly into several individual “rooms” (e.g. using room separation lines).
- Further specification includes the definition of level references, such as floor levels, foundation, intermediate, installation and roof levels.

3.3.3 Level of Development

The Building Information Modeling process at Siemens Real Estate relies on the concept of Level of Detail (LOD) and Level of Information (LOI), to achieve consistency and gradual development of details and attributes throughout the functional element descriptions. For the building elements included in the scope of this work the following minimum requirements regarding the Level of Development have been derived by the BIM@SRE guideline:

LoD of modelling			
Level of Detail	Phases	Data Drop	Description
LoD/LoI 100	Project development Preliminary draft plan	Data Drops 1+2	Rendering with sufficiently accurate outside dimensions
LoD/LoI 200	Schematic design Construction permit	Data Drop 3	Rendering with true dimensions
LoD/LoI 300	Construction plan	Data Drop 4	Rendering with actual structural composition and details
LoD/LoI 400	Installation planning	Data Drop 4	Submission of all details and information for installation
LoD/LoI 500	Construction Documentation	Data Drops 5-7	Update according to as-built status

Figure 3-3 Level of Detail of modeling according to BIM@SRE (Siemens AG Real Estate 2017)

Level of Information (LoI)

According to the LoI, different requirements on functional elements are specified. With the development of the project, the level of required information per object increases progressively, including alphanumerical data and/or supplementary documentation. The following requirements on the level of information, for the early project development (Figure 3-3), have been outlined (Siemens AG Real Estate 2017, p. 62):

- LoI 100: Information, required to extract key values for preliminary draft planning is available. For instance, spaces must include room information, such as room types.
- LoI 200 (builds upon LoI 100): Classifications and information needed for the building permit and the completion of the first design phase are incorporated. Functional elements must include product information, materials and maintenance and/or inspection information for the operation phase

All functional elements must be enriched with additional information for classification. For each project, a matrix, based on these specifications, is to be created, providing parameter name and data type. The key information here is that parameters must be created as Shared Parameters and the associated text file must be included in the deliverables (Siemens AG Real Estate 2017, p. 63).

Parameters for functional element classification			
Parameter name	Requirement	Data type	Example
Room main category	Number and name of the room main category	Text	1. Office
Room subcategory	Number and name of the room subcategory	Text	1.3 Meeting Zone

Figure 3-4 Classification of spaces according to BIM@SRE (Siemens AG Real Estate 2017)

Parameters for functional element classification			
Parameter name	Requirement	Data type	Example
UniClass2015Code (Omniclass if applicable)	Classification number	Text	Pr_40_30_25_42
UniClass2015Title (Omniclass if applicable)	Classification name	Text	Interactive whiteboards

Figure 3-5 Classification parameters (Siemens AG Real Estate 2017)

All rooms (including shafts and all other technical spaces) must be classified according to the SRE main and sub-room category, to allow consistent comparison of the rental percentage of spaces worldwide (Figure 3-4). For all other functional elements, the required classification system is Uniclass2015 or another suitable local taxonomy system. Both parameters for the classification name and number must be populated (Figure 3-5).

Level of Detail (LoD)

The Level of Detail (LoD) provides information about the required geometric modeling accuracy of functional elements for each project phase. For achieving the goals of the elaborated use cases the main requirements on building elements focus on delivering

consistent and semantically enriched models, therefore the level of geometrical maturity would not be further discussed.

3.4 Data Management

The Data Management section provides guidelines on how data is collected, managed, distributed and stored between project participants. Technical requirements are provided to facilitate the collaboration environment, as well as data creation and processing. Along with region-specific definitions for classification, a data submission plan and descriptions of additional databases if applicable, the chapter also specifies the responsibilities for information exchange and describes the used software applications:

- Model building tools - Unless provided otherwise, Autodesk Revit is the model building tool to be used throughout the project. If stakeholders cannot provide a .rvt software format file, the software used to create the affected models should enable efficient exports of IFC2x3 TC1.
- Classification tool – Classification requirements shall be ensured by filling in the required functional element attributes. Classification may be performed using add-in tools (Autodesk Revit add-ins).
- Information transfer tools – additionally to classification, information is to be transported from the model to the operator platform (e.g. Planon) in Cobie XML format.

3.5 Challenges and current practice

The Building Information Modeling process has been established to use digital ways of work to ease collaboration and time management issues in construction projects. Nevertheless, vaguely specified information in guidelines can be wrongly interpreted by project participants, which will lead to cost and time overrun. To ensure consistency and data quality, stakeholders must define standards, that is understandable, specific and usable. A machine-readable guideline can ensure that standards are continuously implemented, and long-form content tasks are divided into smaller work packages for specific parties. BIM@SRE's goals and objectives include (Siemens AG Real Estate 2017, p. 5):

- Consistent implementation of the BIM@SRE standards through our project's lifecycle and proper application of project-specific use cases and their requirements.

- All software applications, formats and interfaces, used to implement the BIM@SRE standards, shall be internationally recognized and well-established in the relevant markets.
- Error-free use of all geometric and alphanumeric data, captured during the planning and construction phases, shall be ensured during operation. Mapping and archiving geometric and alphanumeric data across the entire lifecycle must be possible, without requiring major maintenance efforts. In the event of a remodeling of the building, all documentation must be accessible and reusable at all times.

To encounter these challenging aims the following suggestions shall be considered:

- To assure worldwide recognition, the choice of exchange formats shall primarily consist of open formats. As the IFC format has been continuously developed and improved, the number of software solutions providing IFC compliant interfaces has increased. As part of the ISO standardization, the IFC format has become one of the most widely implemented international standard schemas (buildingSMART International, BLIS Consortium/Richard See 2012).
- To keep data cohesive, one must preferably not divide geometrical from alphanumerical data in the process of modeling. Nevertheless, relying on different propriety formats to exchange information prompts for losing data in the process of work. The IFC schema allows simultaneous representation and storage of geometrical data, enriched with semantic information. However, when remodeling during the operable phase, is necessary, other open formats may serve for collaboration means. The non-proprietary data format COBie focuses on delivering asset data separately form geometric information (Dr Stephen Hamil 2018; Borrmann et al. 2018). It enables information to be extracted from a native BIM model or an IFC file, placed into a standard COBie schema and transferred into a facility management application tool. Still, there are some limitations, such as the demand for consistent information update from a spreadsheet-based deliverable into the IFC file. (Yalcinkaya and Singh 2019). Although COBie deliverables are part of the BIM@SRE requirements, a direct import of spreadsheets into the FM tool Planon is not possible without extensive additional manipulation of the data.

- Since the requirements, provided in the guideline, does not include specifications for particular building objects and their properties, further concretizing will allow a better semantic information enrichment of the models in the future.

4 Methodology

4.1 Use Cases

The definition of quality is multidimensional and based on subjective factors, such as “fit for the purpose”, “free from defects” or even “pleasing to look at” (Tony Cunningham 2013). In terms of BIM data models, quality can be defined as conformance with project standards, that specify information structure and modeling guidelines (Borrmann et al. 2018).

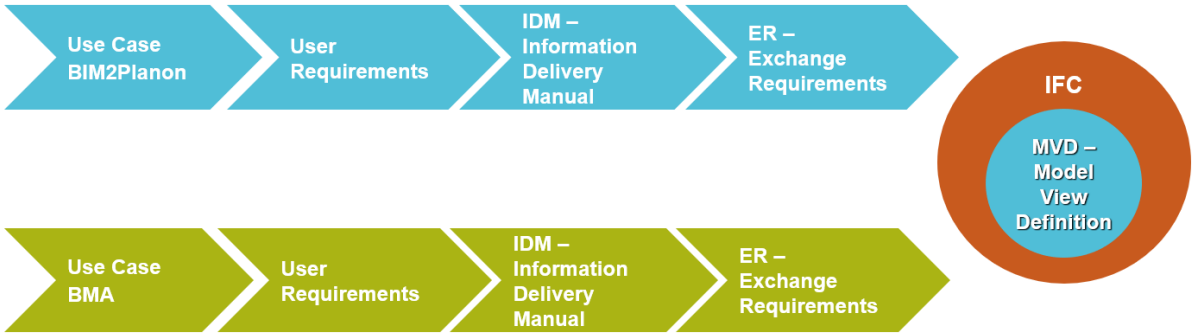

Figure 4-1 Methodology of work

The requirements, outlined upon a detailed study of the Siemens BIM guideline, include primarily qualitative information for objects’ existence and correctness of property values. Given that the BIM@SRE document has more of a guideline character, rather than a regulatory one, there are almost no quantitative requirements or specific value boundaries. For this work and as per the stakeholders needs the following two limitations have been met: (1) developed specifications focus primarily on semantic information of architectural model elements; (2) some steps, part of the IDM approach, such as identifying processes and actors, and defining exact data drops, have not been considered since they have been elaborated in detail in the BIM@SRE guideline.

Based on the needs and interests of Siemens Real Estate and its role as a building owner and investor, further, two use cases: BIM2Planon and Building Model Analytics, have been described (Figure 4-1). Both root from already existing use cases, however, their focus lies strongly on outlining specific exchange requirements and defining deliverables in terms of objects, properties and other concepts.

4.1.1 Building Model Analyzer (BMA)

Nowadays building owners face various challenges in construction projects like low budgets along with constantly increasing prices, late cost certainty and inconsistency in model quality during the planning phase. This results in resource and time overrun, as well as inefficient buildings and late value engineering measures. While construction costs can be influenced efficiently in the early stages of design and tender preparation, value engineering measures and evaluation of technical solutions is performed later, when the design has been finalized. Although the cost of each building is related to the individual design and geometry, benchmarking against similar national and internal projects shows that some of the biggest element groups like walls, façade, windows and doors are also key cost influencers (Christian Schittlich 2012; BKI 2019). Additionally, securing building performance strongly relies on defining and optimizing space efficiency. Implementing the BMA use case in a project shall assure high model quality for early cost and design evaluations and thus provide technical alternatives. Therefore, when implementing this use case in the MVD, the focus lies on defining requirements for building elements to allow automated IFC-based extraction and analysis of space efficiency data and façade cost simulation.

4.1.2 BIM2Planon

Planon is a facility management software, used to streamline assets throughout the building life cycle. The objective of the BIM2Planon use case is to assure, that facility service and facility-related building functional elements are considered and documented. The needs of facility management in the early design project phase are primarily focused on space definitions and room book attributes, that will have a strong impact on operating expenses, once the building is in use. Data requirements for this use case include identification information, classification, geometrical and other semantic data about rooms.

4.2 Investigated concepts

The following concepts have been agreed upon, as main investigated components of the developed Model View Definition. The choice of concepts was primarily based on the BIM@SRE content, however, to meet the goals of the newly outlined use cases, the scope of work was extended.

4.2.1 Classification

As per chapter Data Management, part of the BIM@SRE guideline, the required classification standard for each project is to be chosen between Uniclass2015, for countries such as Germany, and OmniClass e.g. for the USA. Additionally, rooms shall be classified in 21 room and sub-room categories, based on which rentable and non-rentable space is calculated. When defining the exchange requirements, later on, the need for an additional region-specific certification systematic for Germany arises. CAFM-Connect has developed a classification catalog, conformed with the German construction market, consisting of room usage types, according to DIN 277-2 (2005) and component types classified as per DIN 276+ and mapped to IFC (CAFM-Connect).

4.2.2 Material Association

BIM@SRE does not contain specific requirements for values regarding material properties. As stated in the previous chapter, all elements must have material information, when LoI 200 is reached (Siemens AG Real Estate 2017). Material assignments are also required as a deliverable in the last data drop when the “as-built” 3D model is handed over to the building owner Siemens for use in the operational phase (Siemens AG Real Estate 2017)(Siemens AG Real Estate 2017). For this work materials, as one of the cost-driving elements in the Criteria Definition Phase, must be included in the MVD as a checkable concept of each building element.

4.2.3 Element Decomposition

The decomposition of elements is not a concept that has been elaborated upon in the BIM@SRE guidelines since it is mainly supported by the IFC schema. However, it has been included to allow a more cohesive representation and accurate validation of the data model.

4.2.4 Properties and Quantities

Requirements on properties and quantities can be found under various notations, including COBie “Type” Parameters, COBie “Component” Parameters, Cost KPI’s or Building KPI’s, according to their use case allocation. The provided information was filtered, grouped and assigned via the object-based approach to each building element. The choice of categorization criteria was driven by the goal of implementing a unified standardization methodology, based on open format terms and definitions, instead of

fragmenting similar or identical properties in various local-based or project-specific groups.

4.3 Employer's Information Requirements (EIRs)

One of the main challenges when creating EIR's, has been outlining cost-relevant and facility management relevant information for different objects and converting it into a machine-readable format, that will improve representation, data filtration and automated evaluation. Recent studies (Jimmy Abualdenien et al. 2019; Sina Pfuhl October/2018a) and industry's project examples (BLIS Project; STATSBYGG; BLIS Consortium - Digital Alchemy 2019) were examined, compared and used as a baseline for the EIR's and MVD development. MVD specifications in form of mvdXML format, HTML documentation or PDFs are openly available in an MVD Database, provided by bSI (buildingSMART International 2019k).

In the early stages of design, when building elements are not specified in detail, decisions, taken by stakeholders are mainly based on their experience and not on data validation and optimization procedures. Deriving such semantic information is a complex workflow, that requires iterative discussions with discipline experts. Specifying the LoD of objects for each project phase has been another challenge, that brings uncertainty when determining exchange requirements.

Based on the aforementioned research topics and examples, the information requirements provided in BIM@SRE have been first extended and organized in a structured way (Excel file), to later be evaluated and incorporated in the MVD. Afterward, additional steps for technical implementation have been performed. Each of the identified exchange requirements was mapped to a suitable IFC4 schema entity. The domain-specific requirements are structured by Objects (type) e.g. *IfcWall* and include:

- General identification information: *Name*, *Description*, etc. (no check).
- Predefined Types: Additional specification on some Objects (check in the *Applicability* element).
- Property-sets with 'Pset_' prefix: Properties and Property sets part of the IFC base schema - *AcousticRating*, *FireRating*, *IsExternal*, etc. (existence check and value data type and value check if specified).
- User-defined Property with the 'SRE_' prefix: Additional requirements not covered by the IFC Standard.

- Quantities – Requirements on Object (type) *IfcSpace*.

Furthermore, data types and enumeration values from the IFC data schema were mapped to object and property definitions. Additional property sets were created as extensions on the base schema in the case that the IFC data format does not provide definitions like specific manufacturer information or others. During the process of work, results were summarized in a spreadsheet, available in the appendix. Table 4-1 presents a summary of the investigated property set and quantity set definitions, including properties and data types, assigned to IFC entities. The checkmark in the corresponding row indicates, that the property and the respective property set were assigned to this entity definition.

Table 4-1 Definitions of property sets and quantity sets, assigned to IFC entities

Property and data type	**Property sets/Quantity sets and associated IFC entities**																
	IfcSpace				X						X	X	X	X	X	X	
	IfcWall			X						X							X
	IfcCurtain-Wall	X															
	IfcPlate	X							X								X
	IfcMember							X									X
	IfcWindow	X	X				X										X
	IfcDoor	X				X											
		SRE_AssetProperties	SRE_WindowProperties	SRE_WallProperties	Qto_SpaceBaseQuantities	Pset_DoorCommon	Pset_WindowCommon	Pset_MemberCommon	Pset_PlateCommon	Pset_WallCommon	Pset_SpaceCommon	Pset_SpaceLightingRe-	Pset_SpaceOccupancyRe-	Pset_SpaceThermalRe-	Pset_SpaceFireSafe-	Pset_SpaceCovering	Pset_ManufacturerTypeIn-
IsExternal *(IfcBoolean)*						X	X			X							
LoadBearing *(IfcBoolean)*										X							
AcousticRating *(IfcLabel)*							X		X	X							
Fire Rating *(IfcLabel)*						X	X	X	X	X							
ThermalTransmittance (*IfcThermalTransmit-tanceMeasure)*							X	X	X	X							
Compartmentation *(IfcBoolean)*										X							
Glazing Area Fraction *(IfcPositiveRati-oMeasure)*						X	X										
HasDrive *(IfcBoolean)*						X	X										

SecurityRating *(IfcLabel)*						X	X										
ModelReference *(IfcLabel)*																	X
Manufacturer *(IfcLabel)*																	X
ModelLabel *(IfcLabel)*																	X
Reference *Subspace cat. (IfcIdentifier)*											X						
CeilingCovering *(IfcLabel)*																X	
WallCovering *(IfcLabel)*																X	
FloorCovering *(IfcLabel)*																X	
FireExit *(IfcBoolean)*															X		
SpaceHumidityMax *(IfcRatioMeasure)*														X			
SpaceHumidityMax *(IfcRatioMeasure)*														X			
SpaceTemperatureMax *(IfcThermodynamicTemperatureMeasure)*														X			
SpaceTemperatureMin *(IfcThermodynamicTemperatureMeasure)*														X			
NaturalVentilation *(IfcBoolean)*														X			
NaturalVentilationRate *(IfcReal)*														X			
MechanicalVentilationRate *(IfcReal)*														X			
OccupancyNumberPeak *(IfcCount)*													X				
Illuminance *(IfcIlluminance)*												X					
FinishCeilingHeight *(IfcPositiveRatioMeasure)*					X												
FinishFloorHeight *(IfcPositiveRatioMeasure)*					X												
Orientation *(IfcLabel)*		X															
WallAreaDensity *(IfcAreaDensityMeasure)*				X													
HumidityProtection *(IfcBoolean)*				X													
NumberPlankings (*IfcCount)*				X													
PlankingDensity *(IfcLabel)*				X													
PlankingType *(IfcLabel)*				X													
StandAxisDistance *(IfcNonNegativeLengthMeasure)*				X													

SubstructureType *(IfcNonNegativeLength-Measure)*				X													
AttackResistancy *(IfcLabel)*			X														
BlastResistancy *(IfcLabel)*			X														
BulletResistancy *(IfcLabel)*			X														
GlazingType *(IfcLabel)*			X														
SafetyCatcherDesign *(IfcLabel)*			X														
SpacersDesign *(IfcLabel)*			X														
SunShadingDesign *(IfcLabel)*			X														

Currently, model element tables are the basis for defining the required information and LOD for each project stage or milestone. The growing complexity and size of the projects make the maintaining of these spreadsheets time and resource consuming. Moreover, the way information is being represented in these tables, in the form of row and columns, is not compatible with modeling software and model checking software. Thus, restricting project owners and other actors to manually extract information from spreadsheets and PDFs to perform their given task. Proper tools, such as BIMQ (AEC3 2019) and properBIM (BIMtech), can speed up the process and minimize the effort by providing web-based BIM solutions.

4.4 Conceptual approach through IfcDoc

The Model Support Group of BuildingSMART has developed the IFC Documentation Generator to support the generation of Model View Definitions in mvdXML format. The software tool gives access to the overall IFC schema (from IFC4 onwards); supports particular IFC release specifications; provides a user interface for defining mvdXML and additional documentation. IfcDoc provides means to generate concept templates, as well as to further the further defining and specification of the Model Views.

4.4.1 MVD development

The starting point of an MVD development is the fundamental IFC schema specification and the set of reusable MVD concept templates definitions (Chipman et al. 2016, p. 12). BSI provides a baseline, that includes MVD content in the form of reusable concept templates for defining property sets, composition rules, materials and others.

For this master's thesis, the IFC4 Addendum 1 Baseline (with MVD definitions of Reference View and Design Transfer View) has been chosen as a baseline schema. A detailed description of the concept templates, included in the IFC4 Reference View, is provided in Chapter 4 "Fundamental concepts and assumptions" listed in the MVD specification delivery. The process of creating *ConceptTemplates* and the specification of an mvdXML are elaborated below.

Once the schema has been loaded, a Model View with two *Exchange Definitions* for each use case has been created. Afterward, concept templates are set up for each of the objects, defined in the ER's. IfcDoc provides the possibility to assign a Base View to the created MVD, allowing for concept definitions to be inherited from an existing MVD (suitable example here is the Reference View) and to be further modified to align with the newly created "SRE MVD". Concepts, restrictions or extensions can also be manually created and assigned to *ConceptRoots*. *Concepts* can include an indication for import and export requirements for each *ExchangeDefinition*.

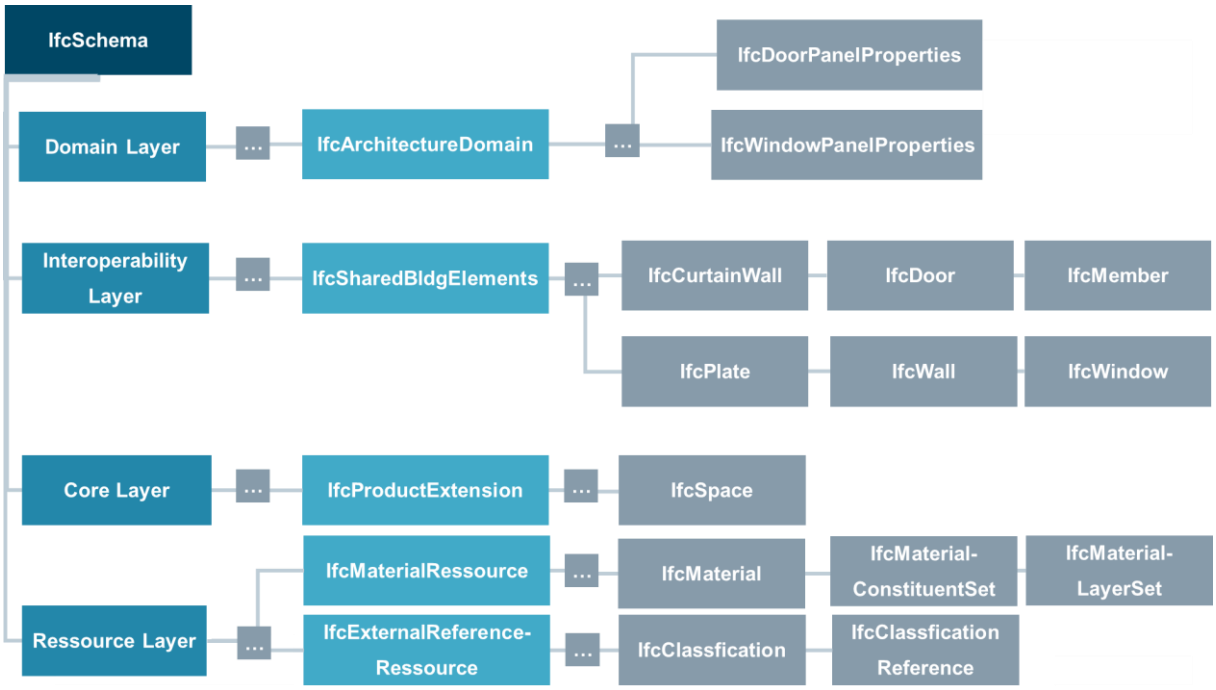

Figure 4-2 Investigated entity definitions in IFC

Figure 4-2 presents the main IFC entity definitions, incorporated in the MVD. The *IfcScharedBldgElemets* schema consolidates entities, representing the main components of a building structure – façade, walls, plates, members, windows and doors. The Core Layer and its *IfcProductExtension* schema provide the spatial project

structure like space definitions (*IfcSpace)* within the project context. Supporting data structures, like *IfcMaterialResource* and *IfcExternalReferenceResource* (location of *IfcClassification*), are part of the Resource Layer.

Figure 4-3 depicts an overview of the explored IFC concepts. Detailed elaboration on the applicable entity definitions, constraints and examples of the configured parameter values is presented below. Since some functionalities are not yet implemented in the IfcDoc12.1 version, possible workarounds are discussed.

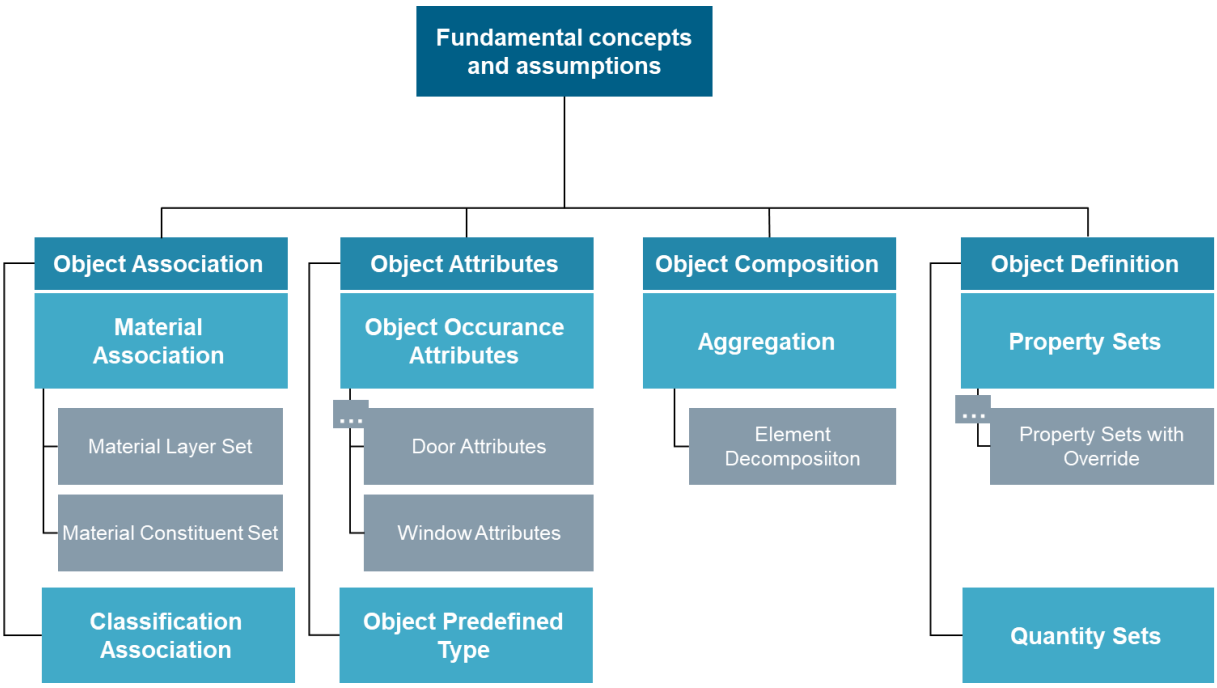

Figure 4-3 Incorporated IFC concepts

Object Predefined Type

IfcSpace, IfcWall, IfcPlate, IfcMember, IfcWindow, IfcDoor

To additionally distinguish entities by particular characteristics, a specific enumeration attribute, named *PredefinedType* is used. The predefined type differentiates objects utilizing enumerators and not subtypes. If a custom value is needed, the *PredefinedType* is set to '-USERDEFINED.' and an attribute *ObjectType* holds the value. If the object is typed, then the *PredefinedType* at the occurrence is to be used only if the PredefinedType at the object type is set to '.NOTDEFINED.'. Table 4-2 describes the applicable predefined types.

Table 4-2 Constraints on the PredefinedType

Component types	Constraints on PredefinedType
Courtyard (IfcSpace)	IfcSpace.PredefinedType='.EXTERNAL.'
Ramp (IfcSpace)	IfcSpace.PredefinedType='.PARKING.'
Parking (IfcSpace)	IfcSpace.PredefinedType='.PARKING.'
Traffic area (IfcSpace)	IfcSpace.PredefinedType='.PARKING.'
Internal spaces (IfcSpace)	IfcSpace.PredefinedType='.INTERNAL.'
Load-bearing External Wall (IfcWall)	IfcWall.PredefinedType='.SOLIDWALL.'
Load-bearing Internal Wall (IfcWall)	IfcWall.PredefinedType='.SOLIDWALL.'
Non-load-bearing Internal Partitioning Wall (IfcWall)	IfcWall.PredefinedType='.PARTITIONING.'
Non-load-bearing Internal Think Thank Glass Wall (IfcWall)	IfcWall.PredefinedType='.MOVABLE.'
Panel (IfcPlate) • Lamella panel • Glass panel • Metal sheet	IfcPlate.PredefinedType='.CURTAIN_PANELL.' IfcPlate.PredefinedType='.CURTAIN_PANELL.' IfcPlate.PredefinedType='.CURTAIN_PANELL.'
Mullion (IfcMember)	IfcMember.PredefinedType='.MULLION.'

Classification Association

IfcSpace, IfcWall, IfcCurtainWall, IfcPlate, IfcMember, IfcWindow, IfcDoor

The ClassificationAssociation concept allows a reference (*IfcClassificationReference*) to an external source of information, such as classification (*IfcClassification*), to be associated to objects and object types (buildingSMART International 2019d). *IfcClassification* is used for the arrangement of objects into a class or category and the attributes *Name*, *Location* and *Source* describe information for the classification system itself, *IfcClassificationReference* gives information about the specific classification key. Here the *Identification* attribute holds the key for the reference to classification items and tables and the *Name* attribute is a human interpretable designation of this key. Each root entity has been assigned the concept *Classification* with a constraint on the attribute IfcClassifcationReference.Name to only consists of "Uniclass Classification" and "CAFM Classification" (Table 4-3).

Table 4-3 Constraints on the IfcClassificationReference.Name

Root entity	Constraints on IfcClassificationReference.Name
IfcSpace	IfcClassificationReference.Name="Uniclass" AND IfcClassificationReference.Name="CAFM"
IfcWall	
IfcCurtainWall	
IfcPlate	
IfcMember	
IfcWindow	
IfcDoor	

Material Association

IfcWall, IfcPlate, IfcMember, IfcWindow, IfcDoor

As specified in the ERs objects' composition must be indicated by an associated single material or set of materials (Table 4-4). In general, materials can have different representations about styles, colors and thermal properties. Materials can be defined by *ConceptTemplates* such as *IfcMaterialSingle*, *IfcMaterialLayer*, *IfcMaterialConstituentSet* and others, or as a fall back by *IfcMaterial* and attached by the *IfcRelAssociatesMaterial* relationship (buildingSMART International 2019j). Since both use cases in the scope of the SRE MVD contain requirements for the early project design, no specific constraints have been defined for *IfcMaterial* parameters' value or syntax. The properties of this concept will be checked only for existence.

Table 4-4 Material association concepts and constraints

Root entity	Material Association concepts and constraints
IfcWall	IfcMaterialMaterialLayer
IfcPlate	IfcMaterialMaterialLayer
IfcMember	IfcMaterialConstituentSet
IfcWindow	IfcMaterialConstituentSet with IfcMaterialConstituent.Name="Lining" IfcMaterialConstituent.Name="Framing" IfcMaterialConstituent.Name="Glazing"

IfcDoor	IfcMaterialConstituentSet

However, some additional constraints were required to assure key information for cost and facility management simulations. For example, the *IfcMaterialConstituentSet,* assigned to *IfcWindow*, must indicate the material of each component (*IfcMaterialConstituents*) – Lining, Framing and Glazing, by populating the value of the *IfcMaterialConstituent.Name.*

The entity *IfcPlate* with a *PredefinedType='.CURTAIN_PANEL.'* represents objects like lamella, glass panels and metal sheets. A human can visually distinguish them by the differences in their appearance, a machine-readable specification, however, must include an unambiguous characteristic to guarantee evaluation and validation. Similarly, a Think Tank Glass wall mapped to *IfcWall.PredefinedType='.MOVABLE.'*, must have an indication of a material that consists of glass.

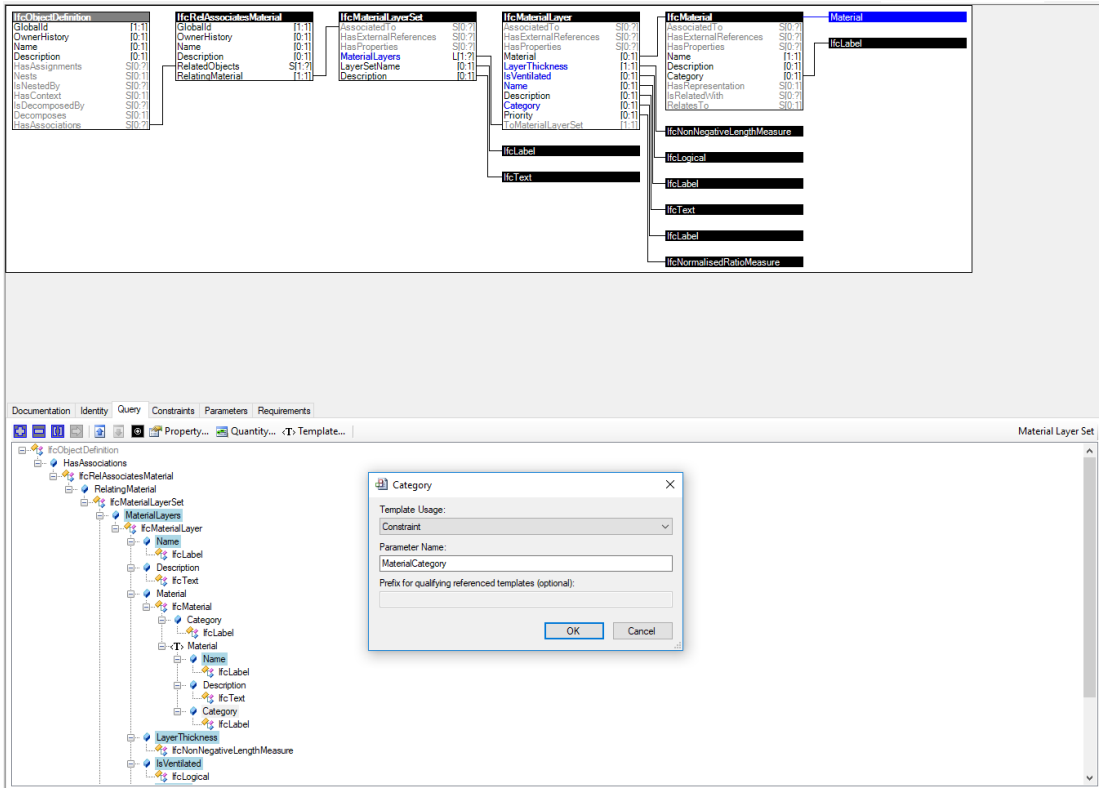

Figure 4-4 MaterialLayerSet concept configuration in IfcDoc

Both entities *IfcMaterialConstituent* and *IfcMaterialLayer* have an attribute *Category*, describing the component's role ('load-bearing', 'thermal insulation', 'inner or outer finish') in a layer set or a constituent set. The assigned *IfcMaterial* also has an attribute

Category, holding information about the general category of the material, such as “Steel” or “Glass” (buildingSMART International 2019c). To create a checkable rule, a new parameter with the name *MaterialCategory* has been created as a constraint of the attribute *IfcMaterial.Category* (Figure 4-4). However, IfcDoc does not provide a user interface to configure a specific value, without changing the *ConceptTemplate*, so this has been adjusted manually in the generated mvdXML.

Property Sets with Override

IfcSpace, IfcWall, IfcCurtainWall, IfcPlate, IfcMember, IfcWindow, IfcDoor

The concept template *Property Sets for Objects* describes how an object occurrence can be related to property sets with multiple properties. The *concept Property sets with Override* extends this concept by allowing a property to be either mapped to the individual object, or the object type. If this property is provided by the same property set then the property, directly assigned to the object occurrence, overrides the property assigned to the object type (buildingSMART International 2019g).

IFC4 ADD2 TC1 supports 160 Property sets, that contain properties, grouped by a common theme. Each property has a name, a description and a value of a given datatype. User-defined extensions can be made to the property definitions, however, the name prefix “Pset_" is intended only for properties, defined within the original scope of the standard (buildingSMART International 2019i). A detailed list of all applied and created properties and property sets is available in the appendix. Further constraints on properties have been defined to support better IFC data filtering and validation (Table 4-5). Constraints’ values cannot be specified in IfcDoc, thus they have been added manually to the generated mvdXML.

Table 4-5 Constraints on properties

Component types	Constraints on properties
Ramp IfcSpace.PredefinedType='.PARKING.'	Pset_SpaceCommon.Reference="8.1"
Parking IfcSpace.PredefinedType='.PARKING.'	Pset_SpaceCommon.Reference="8.1"
Traffic area IfcSpace.PredefinedType='.PARKING.'	Pset_SpaceCommon.Reference="8.1"
Load-bearing External Wall IfcWall.PredefinedType='.SOLIDWALL.'	Pset_WallCommon.IsExternal=TRUE Pset_WallCommon.LoadBearing=TRUE
Load-bearing Internal Wall IfcWall.PredefinedType='.SOLIDWALL.'	Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=TRUE
Non-load-bearing Internal Partitioning Wall IfcWall.PredefinedType='.PARTITIONING.'	Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=FALSE
Non-load-bearing Internal Think Thank Wall IfcWall.PredefinedType='.MOVABLE.'	Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=FALSE

Quantity Sets

IfcSpace

Similar to the concept *Property set for Objects*, objects can be also referenced by quantity sets, containing multiple quantity occurrences, with data types, giving information about length, area, volume, weight, time. Each quantity is defined by its name and value, and optionally a description and a formula. Some geometric information is automatically exported by common design software tools, certified for IFC4 RV export. Furthermore, both use cases do not require specific or detailed geometrical information outside of this scope. Thus, the concept has been assigned solitary to IfcSpace, to provide information about the height of the finished ceiling and floor (Table 4-6).

Table 4-6 QuantitySet concept

Root entity	Quantity Sets concept
IfcSpace	Qto_SpaceBaseQuantities.FinishCeilingHeight Qto_SpaceBaseQuantities.FinishFloorHeight

Element Decomposition

IfcCurtainWall

The concept *ElementDecomposition* provides an aggregation structure, where a composite element is decomposed by other elements into parts. The composite shall not have own Body Geometry and Material assignment since they are provided by the parts. The concept template is assigned to the *IfcCurtainWall* root entity, given that the façade, is composed of the elements *IfcPlate and IfcMember*. Accordingly, the attribute of the relationship entity *IfcRelAggregates* is configured to include the two entities (Table 4-7).

Table 4-7 ElementDecomposition concept and parameter constraints

Root entity	Element Decomposition elements
IfcCurtainWall	IfcPlate.PredefinedType='.CURTAIN_PANELL.'
	IfcMember.PredefinedType='.MULLION.'

Window and Door Attributes

IfcWindow and IfcDoor

Specific entities have additional attributes defined to describe common characteristics at occurrences (buildingSMART International 2019f). Additional semantic and cost relevant information for the operational type of window and door panels can be provided by the enumerator attributes *PanelOperation (IfcDoorPanelProperties*) and *OperationType* (*IfcWindowPanelProperties*), part of the concepts *IfcDoorAttributes* and *IfcWindowAttributes* respectively.

4.4.2 Documentation and manual customization of the mvdXML

IfcDoc provides means to develop and generate MVD concept definitions in the official mvdXML1.1 format, as well as the mvdXML1.2 format. Since mvdXML1.1 has limited support on nested views, parameters and requirements, the focus of this work lies in generating an MVD in mvdXML1.2 and manually enriching it with specifications, that were not able to be implemented by the tool. In following the observed limitations and manually provided adjustments are discussed:

Adding rules in the Applicability element.

One of the set objectives for this work has been implementing constraints on specific entities, that need to be validated before the associated concepts are checked. For example, walls from the type “Load-bearing External Wall”, according to Siemens specifications, are differentiated from other walls by the following constraints IfcWall.PredefinedType=‘.SOLIDWALL.‘; Pset_WallCommon.IsExternal=TRUE and Pset_WallCommon.LoadBearing=TRUE. Such constraints are defined as a list of *TemplateRule* in the *Applicability* element. Table 4-8 summarizes the constraints and their values for different entity instances.

Table 4-8 Externed list of constraints

Building objects	An extended list of constraints
Courtyard	IfcSpace.PredefinedType=‘.EXTERNAL.‘
Ramp	IfcSpace.PredefinedType=‘.PARKING.‘ Pset_SpaceCommon.Reference=“8.1”
Parking	IfcSpace.PredefinedType=‘.PARKING.‘ Pset_SpaceCommon.Reference=“8.1”
Traffic area	IfcSpace.PredefinedType=‘.PARKING.‘ Pset_SpaceCommon.Reference=“8.1”
Load-bearing External Wall	IfcWall.PredefinedType=‘.SOLIDWALL.‘ Pset_WallCommon.IsExternal=TRUE Pset_WallCommon.LoadBearing=TRUE
Load-bearing Internal Wall	IfcWall.PredefinedType=‘.SOLIDWALL.‘ Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=TRUE
Non-load-bearing Internal Partitioning Wall	IfcWall.PredefinedType=‘.PARTITIONING.‘ Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=FALSE
Non-load-bearing Internal Think Thank Wall	IfcWall.PredefinedType=‘.MOVABLE.‘ Pset_WallCommon.IsExternal=FALSE Pset_WallCommon.LoadBearing=FALSE

After generating the mvdXML an inconsistency in the *Applicability* element has been observed: the referenced Concept Template is not existing in the document, meaning that the specified parameters and values in the *TemplateRule* will not be validated.

```
        <ConceptRoot uuid="00e4509a-f99a-475d-b3eb-391688bfcb61" name="Load
bearing External Wall" status="sample" applicableRootEntity="IfcWall">
          <Definitions>
            <Definition>
              <Body><![CDATA[Specification for an external load-bearing
wall with PredefinedType SOLIDWALL.]]></Body>
            </Definition>
          </Definitions>
          <Applicability uuid="00000000-0000-0000-0000-000000000000"
status="sample">
            <Template ref="35d6cef3-c9da-4437-bd7c-c68876fdde2a" />
              <TemplateRules operator="and">
```

Listing 4-1 Template reference mistake in Applicability element

After adjusting the MVD once more in IfcDoc: both concepts *Object PredefinedType* and *Property Sets with Override* were assigned as concepts in the *Concept* element. After generating the mvdXML, the aforementioned constraints were copied and manually adjusted in the *Applicability element*. However, the structure of the mvdXML format allows to reference only one *ConceptTemplate*, therefore the list of *TemplateRules* cannot contain a combination of rules, with parameters, contained in multiple *ConceptTemplates.*

```
        <ConceptRoot uuid="00e4509a-f99a-475d-b3eb-391688bfcb61" name="Load
bearing External Wall" status="sample" applicableRootEntity="IfcWall">
          <Definitions>
            <Definition>
              <Body><![CDATA[Specification for an external load-bearing
wall with PredefinedType SOLIDWALL.]]></Body>
            </Definition>
          </Definitions>
          <Applicability uuid="00000000-0000-0000-0000-000000000000"
status="sample">
            <Template ref="35d6cef3-c9da-4437-bd7c-c68876fdde2a" />
              <TemplateRules operator="and">
              <TemplateRule Parameters="PredefinedType[Value]='SOLIDWALL'
AND TypePredefinedType[Value]='SOLIDWALL'" />
              </TemplateRules>
          </Applicability>
```

Listing 4-2 Constraints on PredefinedType in the Applicability element

Alternatively, the Applicability element may impose that only *IfcWall* instances, that have the two properties *IsExternal* and *LoadBearing* with specific values assigned, must further be checked.

```
        <ConceptRoot uuid="00e4509a-f99a-475d-b3eb-391688bfcb61" name="Load
bearing External Wall" status="sample" applicableRootEntity="IfcWall">
          <Definitions>
            <Definition>
              <Body><![CDATA[Specification for an external load-bearing
wall with properties LoadBearing and IsExternal.]]></Body>
            </Definition>
          </Definitions>
          <Applicability uuid="00000000-0000-0000-0000-000000000000"
status="sample">
              <Template ref="e26040e8-82e2-4f6a-bc63-ac8e6da2d0ae" />
             <TemplateRules operator="and">
             <TemplateRules operator="or">
             <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='LoadBearing' AND Value[Value]=TRUE" />
             <TemplateRules operator="and">
             <TemplateRule
Parameters="TypePsetName[Value]='Pset_WallCommon' AND
TypePropertyName[Value]='LoadBearing' AND TypeValue[Value]=TRUE" />
              <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='LoadBearing' AND Value[EXISTS]=FALSE" />
             </TemplateRules>
              </TemplateRules>
              </TemplateRules>
             <TemplateRules operator="and">
             <TemplateRules operator="or">
             <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='IsExternal' AND Value[Value]=TRUE" />
             <TemplateRules operator="and">
             <TemplateRule
Parameters="TypePsetName[Value]=''Pset_WallCommon' AND
TypePropertyName[Value]='IsExternal' AND TypeValue[Value]=TRUE" />
              <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='IsExternal' AND Value[EXISTS]=FALSE" />
             </TemplateRules>
             </TemplateRules>
             </TemplateRules>
             </TemplateRules>
          </Applicability>
```

Listing 4-3 Constraints on properties in the Applicability element

Inheriting concepts from add-on views

Inheriting concepts form an add-view, that has been selected, is an approach, that requires less manual configuration and thus is prone to fewer mistakes. However, this functionality in IfcDoc has some limitations. By adding a new property, an empty row was created, which resulted in an ‘empty’ *TemplateRule* snippet in the generated mvdXML. The tool prohibits from deleting the empty rows; thus, they must be manually removed from the mvdXML. To minimize such customized adjustments, the properties have been created one by one for each property set in IfcDoc (Figure 4-5). Consequentially this resulted in the segregation of each property in a single *TemplateRule* snippet in the mvdXML.

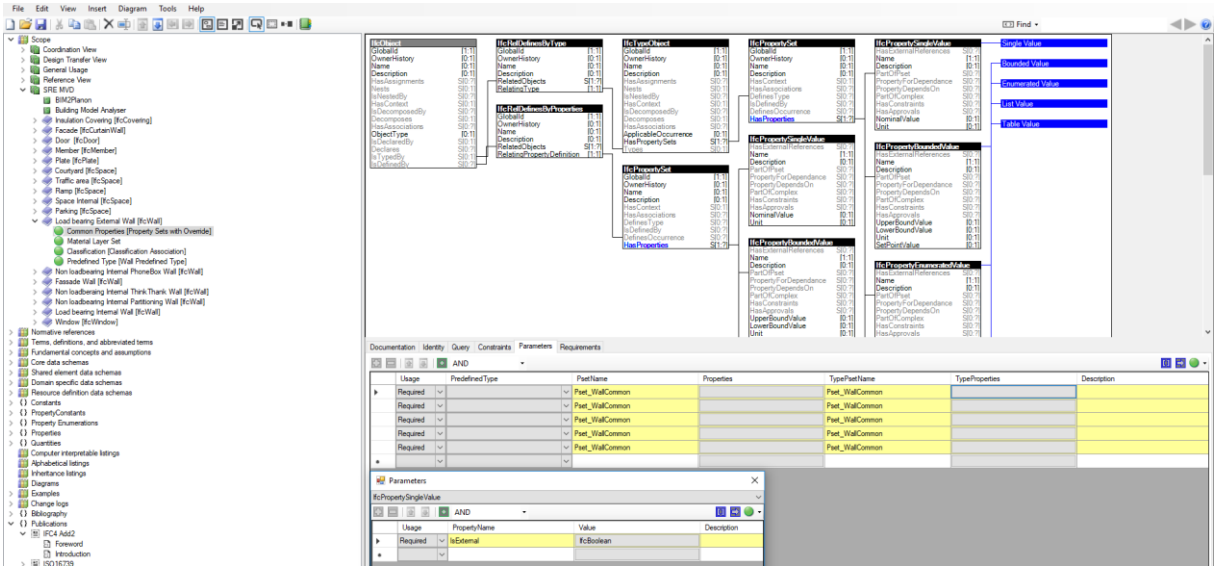

Figure 4-5 Property sets with Override concept in IfcDoc

Extending content to allow further validation of property existence, values and valid syntax.

Since some functionalities of IfcDoc are limited, all concepts have been manually adjusted to align with the content of the predefined exchange requirements. For example, additional *TemplateRules* and parameters have been created to allow the further specification of values and their syntax. Some concepts have been modified to allow validation of property values and valid syntax (e.g. *PropertySets with Override*), whereas others are to be checked only upon existence since their input is highly depending on local requirements and designer choices (e.g. *Material Association*).

```
            <Concept uuid="b129889b-0742-4bb8-89c8-e6d1190bc893"
name="Material Layer Set" status="sample" override="false">
              <Template ref="dfb5b3a9-50e3-46e2-8518-1f03e7cfd886" />
              <Requirements>
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="7fab58cb-4655-489a-862d-85e26a3096b2" />
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="cfa72802-9d70-407c-9616-56f5b53f8708" />
              </Requirements>
             <TemplateRules operator="and">
               <TemplateRule xsi:type="TemplateItem"
Parameters="HasLayer[Exists]=TRUE">
                 <References />
               </TemplateRule>
               <TemplateRule xsi:type="TemplateItem"
Parameters="LayerName[Exists]=TRUE">
                 <References />
               </TemplateRule>
               <TemplateRule xsi:type="TemplateItem"
Parameters="LayerThickness[Exists]=TRUE">
                 <References />
               </TemplateRule>
               <TemplateRule xsi:type="TemplateItem"
Parameters="Category[Value]='Glass'">
                 <References />
               </TemplateRule>
               <TemplateRule xsi:type="TemplateItem"
Parameters="Category[Value]='Metal Sheet'">
                 <References />
               </TemplateRule>
               <TemplateRule xsi:type="TemplateItem"
Parameters="Category[Value]='Lamella'">
                 <References />
               </TemplateRule>
             </TemplateRules>
           </Concept>
```

Listing 4-4 Constraints on MaterialLayerSet parameters

Additionally, it was not possible to represent a number of desired requirements, for instance, the complexity of a check, such as the amount of glass area of a window, positioned in a wall, part of a specific room. Another example is the evaluation of wall or ceiling coverings in a room. In the developed MVD the assumption was made, that this type of information will be represented as a property of the *IfcSpace* entity since it is already part of the property set *Pset_SpaceCoveringRequirements* and no objects of the type IfcCovering were examined. However, independent from the chosen way of implementation, the type of different wall coverings in a space cannot be derived as a separate piece of information, part of a room book. For the building owner to extract this data, another method for linking ERs to MVD should be considered.

4.4.3 Summary

In summary, the IfcDoc tool was created to develop and documentation IFC releases. Although it has some limitations and crucial functionalities have not yet been implemented, it allows a consistent computer-interpretable definition of Model View Definitions (MVD) as subsets of the IFC schema specification (buildingSMART International 2019h). To encourage further development and implementation of the IFC format, as well as an open and free of cost exchange of structured information, the functionalities of the tool need to be updated to match the users' demand.

4.5 Conceptual approach through BIMQ

BIMQ is a web-based platform, developed by AEC3, offering a method to optimize, standardize and automate the processes of exchange requirements creation. The tool provides ways to structure project information requirements and define who delivers what information at which point and in what form. The project guidelines can contain geometric requirements in both written form as well as constraints on geometric representations, and semantic information of element required data format for attributes and properties (AEC3 2019). BIMQ offers different export deliverables depending on the project actor's needs. In addition to generating detailed information requirements in a PDF form, the tool also creates project templates for software like ArchiCAD or Revit and export validation rules as mvdXML1.1 and in Excel format for configuration of Solibri ruleset SOL/203 (Figure 4-6).

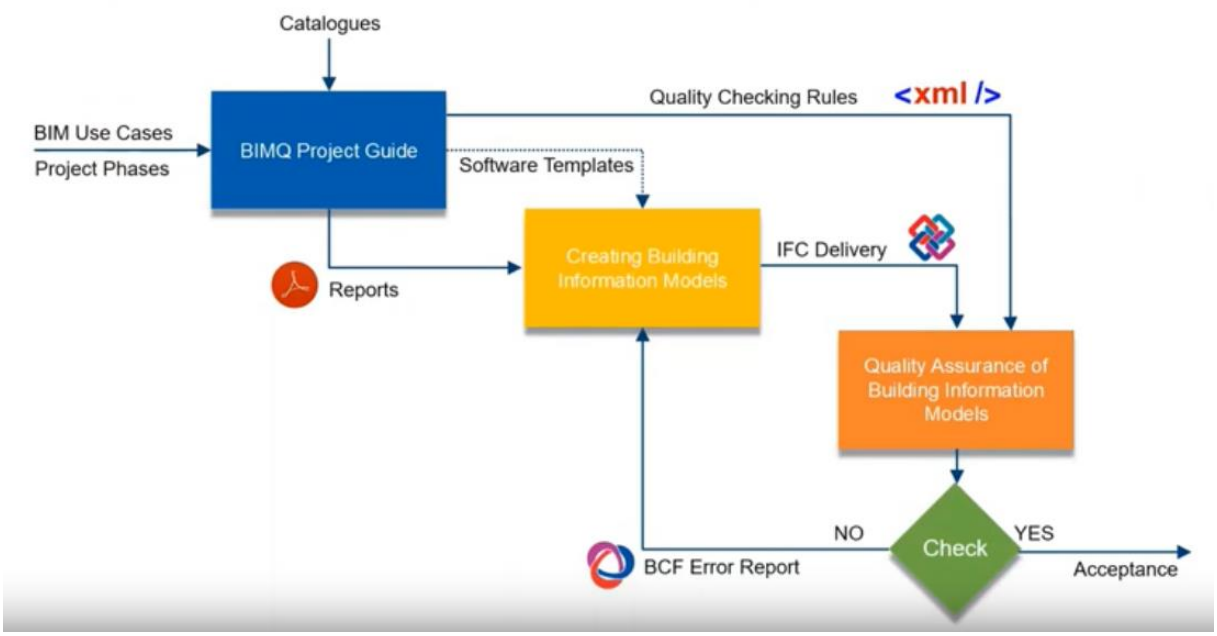

Figure 4-6 BIMQ utilization of mvdXML (BIMQ 2018)

4.5.1 Information management

The development of a digital project guideline starts with creating a project via a template, provided by BIMQ. Afterward, the use cases are defined and assigned to a building work phase. Since exchange requirements are required in the early stages of the project, the use cases are to be implemented by an architectural planner in the Preliminary design phase and the Final design phase. According to the BIMQ platform, the Preliminary design phase shall include a definition of spaces and initialization of

considerations on the basic project elements. As a basis for further design, project spatial and element criteria are fully defined in the Final design phase ((AEC3 2019).

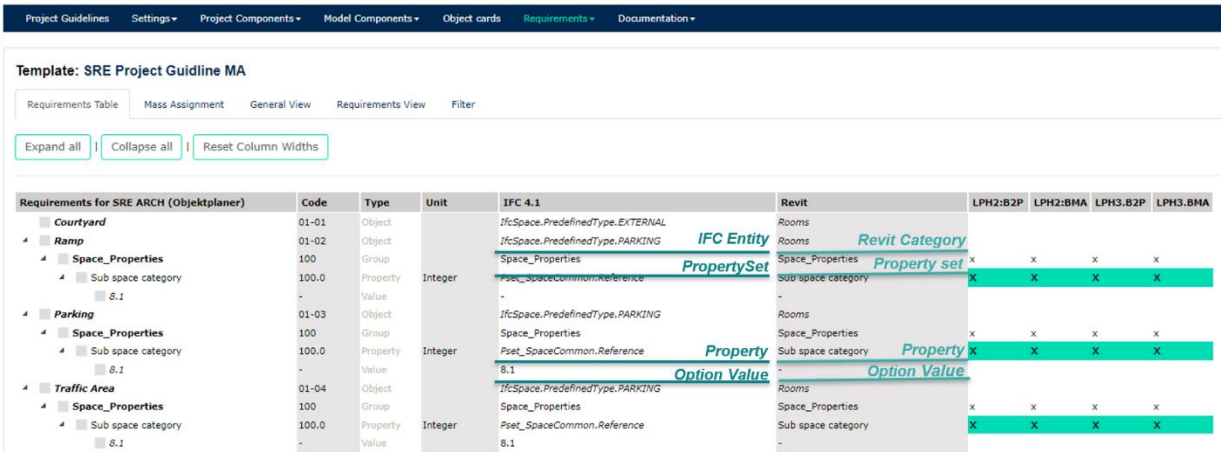

Figure 4-7 BIMQ overview

The created digital Siemens guideline (SRE Project Guideline) consists of an architectural discipline model, model elements, properties, quantities and IFC concepts (Figure 4-7). After setting up the project, each of the listed components is enriched with semantic information from the ER's Excel spreadsheet. Initially, each information piece - element, property set, a single property, etc., must be individually created and enriched with information like the unit group, unit type and preferably the unit itself. However, once defined, these concepts can be easily assigned to multiple elements and project templates.

The user interface allows easy and intuitive data mapping to IFC4 or IFC2x3 entities, property sets, and single properties. Alternatively, BIMQ also provides an import feature for mvdXML concept templates, allowing to configure other IFC concepts such as Classification and Material Association, Element Decomposition and others (Figure 4-8). An example of how *TemplateRules* is configured, via snippets as XML-Code is provided for each of the created IFC concepts. The full mvdXML document is provided in the appendix.

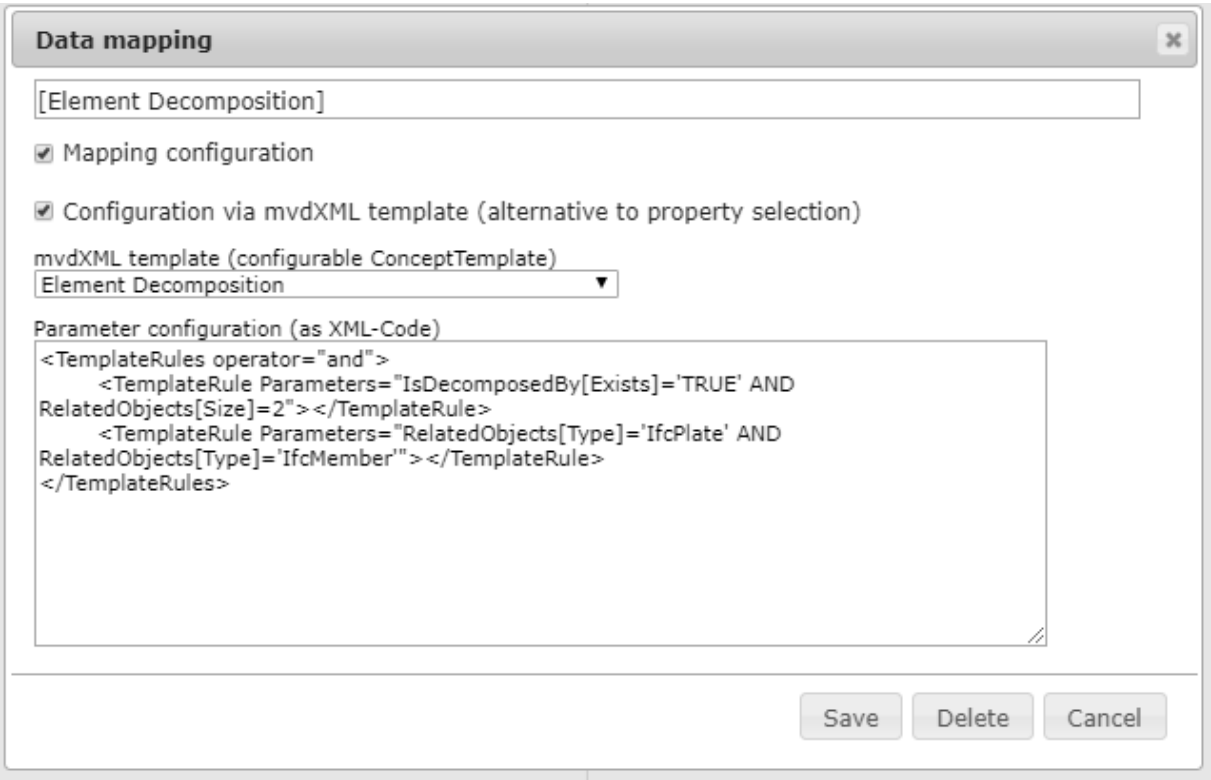

Figure 4-8 TemplateRules configuration snippet in BIMQ

BIMQ also allows mapping entities and properties to the corresponding Revit categories. Revit can automatically assign default attributes or properties to units of information in IFC, ensuring that the necessary information is being exchanged cohesively (Autodesk Inc 2018). These attributes and properties are automatically recognized and assigned correctly by the IFC export tool, as soon as they are populated with a value. Table 4-9 gives an overview of the default attributes provided by Revit, that are of interest for the scope of the work, and the correspondent IFC properties.

Table 4-9 IFC Properties and correspondent default parameters in Revit

IFC properties	Default attributes in Revit
Reference	Component type (type name)
FireRating	Fire-resistance class (type parameter)
ThermalTransmittance	U-value (type parameter)
IsExternal	Exterior component (type parameter, given as Boolean value)
LoadBearing	Load-bearing (instance parameter)
Model	Model type of the element
Manufacturer	Manufacturer of the element materials

4.5.2 Documentation and manual customization of the mvdXML

Once exchange data is provided in a structured manner, associated documentation and software templates can be generated. Detailed information requirements of model elements and properties, according to project phases can be exported in PDF format. To ensure interoperability and enable further data validation BIMQ can export different application formats and support integration with common BIM tools.

From the SRE Project guideline a shared parameters file for Revit and an IFC mapping file, both in text format (.txt), have been used to transfer the defined ERs to Revit. Additionally, validation rules were generated in mvdXML1.1 format (.mvdXML) and as an Excel file for the configuration of rulesets in the Solibri Model Checker. When generating the mvdXML tool limitations can be overcome to an extent, by manually implementing changes via snippets directly in the online platform. All content modifications performed on the mvdXML, generated in IfcDoc (see Chapter 4.4.2) have also been applied to the mvdXML, exported by BIMQ, to allow a consistent validation workflow.

4.5.3 Summary

BIMQ allows the preparation of information requirements according to performance specifications and working phases. The project-specific guidelines and the direct derivation of check rules in mvdXML format or in Excel format allow for the building owner to automatically validate the submitted discipline-based models against information requirements. Developing an mvdXML using BIMQ’s interface did not reveal any setbacks, as users can manually configure each concept using the provided snippets.

5 Case Study

5.1 Scope

To validate the applicability and correctness of the developed MVDs and perform model checks, an example office building was modeled in Revit 2019.2. The model was created to assure that the correct modeling techniques are applied and the complete definition of the component-specific parameters in Revit is included. While defining a Model View Definition with a high level of data richness and specificity, it is important to consider how much of this information can be created with the available software tools on the market. Next to creating the model, the specific IFC Export configurations are crucial for the actual output, as well. A brief explanation of the IFC file structure and the required export setting is presented below.

An IFC file consists of a header and a body. General information about the building model used software and the IFC version is displayed in the header, while the body contains geometric and semantic data of the building elements. Element description begins with a line, containing the IFC entity classification, the universally unique identifier and optional information like owner history, name and description. Information about the position sequence of each additional attribute can be found in the IFC schema specifications on the bSI technical web page (buildingSMART International 2019d).

#11759= IFCWINDOW('**1vMgHDnNT8MfGaLIJlDleb**',#42,'FE T\X2\00DC\X0\R 1 tlg - Fix-1 Nurglasecke:1100 x 2250:2407508',$,'1100 x 2250'**,#116578,#11752**,'2407508',2.47937475657232,1.37527819818117,.WINDOW.,.NOTDEFINED.,$);

#11677= IFCPRODUCTDEFINITIONSHAPE($,$,(#11675));

#115002= IFCLOCALPLACEMENT(#114990,#115001);

Listing 5-1 IFC file structure

The structure of the IFC file allows linking further concepts to this entity until a logical model has been generated to provide the complete information available for the object. This method also guarantees that particular information sets (e,g, materials) are stored only once, but can be referenced multiple times (Autodesk Inc 2018).

The following settings by the IFC export have been modified, (Figure 5-1):

- General:
  - IFC version: IFC4 Reference View.
  - Space boundaries: 1st Level.
  - Split elements like Walls and Columns by level.
- Additional Content: Export rooms in 3D views.
- Property Sets: Export user-defined property sets: linked to custom parameter sets text file, generated my BIMQ.
- Level of Detail: no additional settings.
- Advanced: Store the IFC GUID in an element parameter after export.

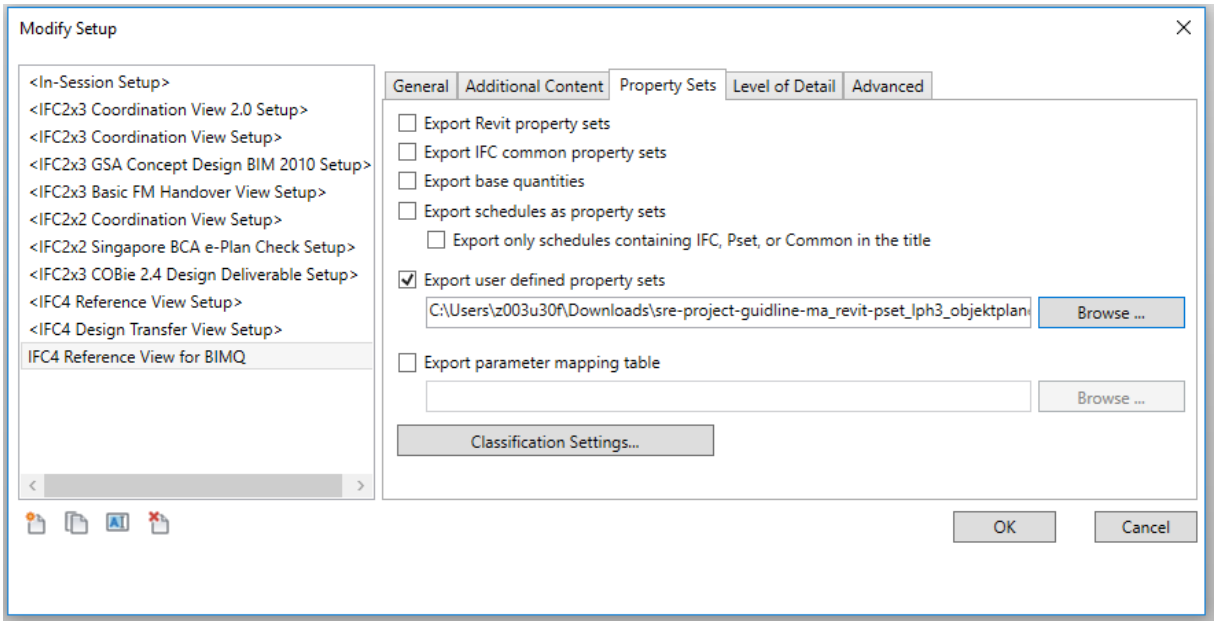

Figure 5-1 IFC export settings in Revit

5.2 Implementation of Concepts

The following chapter traces the steps to create and export the desired information in an IFC file.

5.2.1 Classification

The Classification manager in Revit offers the possibility to set up and assign classifications from a predefined list (Uniclass 2015, OminClass Database, IFC4 Add1 Databases and others) or create a unique classification. According to BIM@SRE, the main

classification used shall be the Uniclass 2015 terminology. When assigning classification through the Classification Manager, Revit automatically creates the following project parameters (Emma Hooper):

- Classification.Uniclass.Ss.Number,
- Classification.Uniclass.Ss.Description
- Classification.Uniclass.EF.Number
- Classification.Uniclass.EF.Description
- Classification.Uniclass.Pr.Number
- Classification.Uniclass.Pr.Description

A primary classification system has to be set through the IFC exporter, using the Classification Settings. Populating the *Classification field name* with the project parameters, created by the Classification Manager, guarantees a correct mapping of the *IfcClassificationReference* (Listing 5-2). Without this step, the classification number and description are not mapped to the *IfcClassificationRefrence*, but exported as properties, part of the 'Data' Property set.

#4553**= IFCCLASSIFICATION**('NBS','2015',$,'**Uniclass**',$,$,$);

#4556=
IFCCLASSIFICATIONREFERENCE('https://toolkit.thenbs.com/Definitions','**Ss_25_10'**,$,#4553,$,$);

#4561= IFCRELASSOCIATESCLASSIFICATION('2jtcXk41f3PQQKKj_Gwk7_',#42,'Uniclass Classification','',(#4490),#4556);

#4565= **IFCCLASSIFICATIONREFERENCE**('https://toolkit.thenbs.com/Definitions',**'Framed wall systems'**,$,#4553,$,$);

Listing 5-2 Classification information in Revit

Additionally to the classification, required by BIM@SRE, a project in Germany can have a regional-based additional classification, such as classification for room usage types (DIN277-2) and component types (DIN276+). For secondary classification systems, Autodesk provides up to ten shared parameters, called *ClassificationCode*. Classification information can be populated with the following syntax –"[ClassificationName] Code: Title". As an example (Listing 5-3), the BIM profile CAFM-Connect classification (CAFM-Connect) was used to assign a classification code and a title ("[CAFM]331:Tragende Aussenwaende") to a wall object

#6077= **IFCCLASSIFICATION**('','',$,'**CAFM**',$,$,$);

#6078= **IFCCLASSIFICATIONREFERENCE**($,'**331**','**Tragende Aussenwaende**',#6077,$,$);

#6080= IFCRELASSOCIATESCLASSIFICATION('2vVZHe8qX3GxuJ6DUytzFv',#42,'CAFM Classification','',(#6050),#6078);

Listing 5-3 Additional classification information in IFC

Another option to assign classification is through the type properties Assembly Code and Assembly Description, as well as the Keynote type property. Both the Assembly code and Keynote's content is modifiable through a text file. Nevertheless, currently, it is not possible to change the predefined classification name- Unifromat Classification, to any user-defined classification name, thus making this approach not suitable for proper classification.

5.2.2 Material Association

After assigning materials to each of the relevant functional elements (walls, plates, mullions, windows, and doors), the exported IFC file is checked for consistency. Listing 5-4 presents an *IfcWindow* entity and the assigned material concept *IfcMaterial ConstituentSet*, that contains two material constituents.

#11759= IFCWINDOW('1vMgHDnNT8MfGaLIJlDleb',#42,'FE T\X2\00DC\X0\R 1 tlg - Fix-1 Nurglasecke:1100 x 2250:2407508',$,'1100 x 2250',#116578,#11752,'2407508',2.47937475657232,1.37527819818117,.WINDOW.,.NOTDEFINED.,$);

*#11762= IFCMATERIALCONSTITUENT('Kunststoff - grau 70-70-70',$,#10832,$,**'Materials'**);*

*#11763= IFCMATERIALCONSTITUENT('Glas - Isolierverglasung klar',$,#10816,$,**'Materials'**);*

#11764= IFCMATERIALCONSTITUENTSET('MaterialConstituentSet',$,(#11762,#11763));

…

*#10816= IFCMATERIAL('Glas - Isolierverglasung klar',$,**$**);*

*#10832= IFCMATERIAL('Kunststoff - grau 70-70-70',$,**$**);*

Listing 5-4 Material information in IFC

The first attribute of the entity *IfcMaterialConstient* is expected to be populated by value, indicating the material constituent name as 'Lining', 'Framing' or 'Glazing'. However, here the exported value specifies the material itself. Similar mistakes were

identified by the attributes *IfcMaterialConstiuent.Category* and *IfcMaterial.Category*. As seen in Listing 5-5 the attribute *IfcMaterialLayer.Category* is not populated (‘$’).

#31408= IFCWALL('14dgzb9KD9EOWiSv7OROTV',#42,'Basic Wall:STB 250:2423247',$,'Basic Wall:STB 250:712295',#31302,#31404,'2423247',.SOLIDWALL.);

#31411= IFCMATERIALLAYERSETUSAGE(#4525,.AXIS2.,.NEGATIVE.,0.125,$);

#4525= IFCMATERIALLAYERSET((#4523),'Basic Wall:STB 250',$);

*#4523= IFCMATERIALLAYER(#4505,0.25,$,$,$,**$**,$);*

*#4505= IFCMATERIAL('Ortbeton - bewehrt Verputzt',$,**$**);*

Listing 5-5 Material information of an IfcWall entity

A possible explanation for this erroneous export could be that the attribute *Category* was first added in the IFC4 schema version. Since Autodesk Revit IFC4 certification is still in progress, the export of this attribute is not supported.

5.2.3 Properties and Quantities

To create the outlined exchange requirements as properties in Revit, a shared parameters file and an IFC mapping file are required. Once the share parameter file is imported, an additional step requires to manually assign the properties to the correspondent Revit category (Figure 5-2).

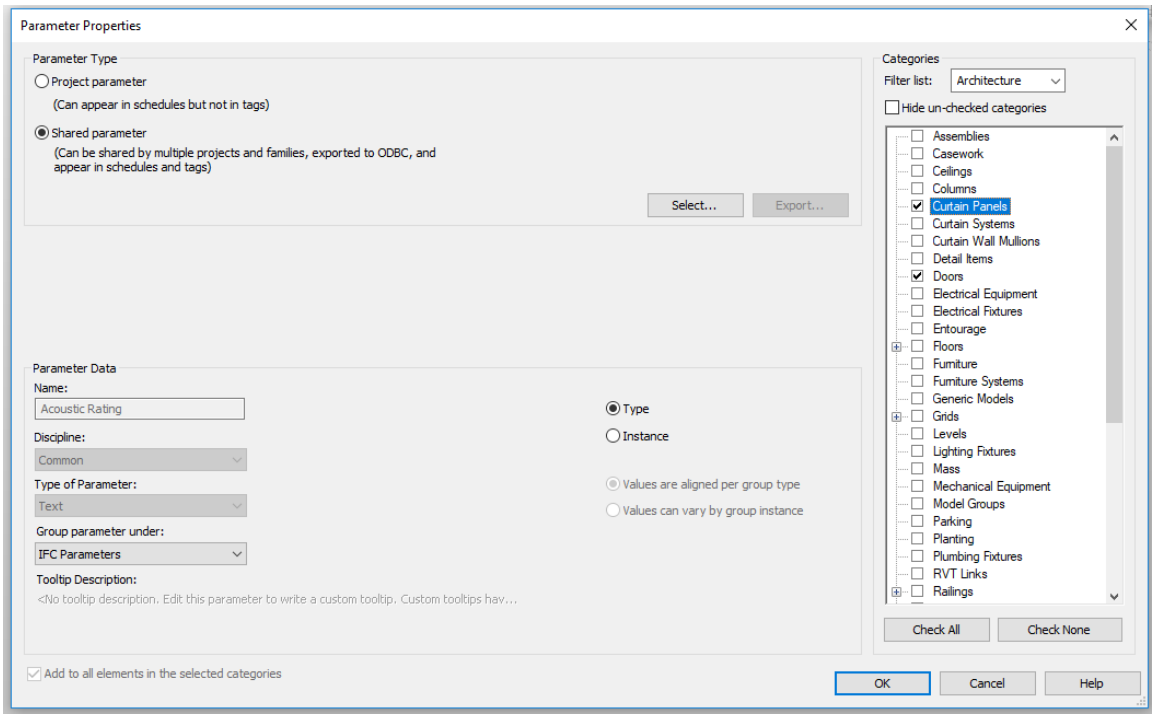

Figure 5-2 Parameter properties in Revit

Mapping Revit categories to the correct IFC class ensures that the full extent of information is exported correctly (Listing 5-6). To export type information (*PredefinedType*), the parameters *IfcExportAs* and *IfcExportType*, that override the element's class, need to be assigned.

*#**4490**= IFCWALL('2_CB1W$Bz6ihhQhSVugWwu',#42,'Basic Wall:STB 250:2398444',$,'Basic Wall:STB 250:712295',#4303,#4486,'2398444',.**SOLIDWALL.);***

…

*#**4533**= IFCPROPERTYSINGLEVALUE('Reference',$,IFCIDENTIFIER(**'STB 250'**),$);*

*#**4534**= IFCPROPERTYSINGLEVALUE('LoadBearing',$,IFCBOOLEAN(.**T**.),$);*

*#**4535**= IFCPROPERTYSINGLEVALUE('ExtendToStructure',$,IFCBOOLEAN(.**T**.),$);*

*#**4536**= IFCPROPERTYSINGLEVALUE('IsExternal',$,IFCBOOLEAN(.**T**.),$);*

…

*#**4540**= IFCPROPERTYSET('2_CB1W$Bz6ihhQfZhugWwu',#42,'**Pset_WallCommon**',$,(#**4533,#4534,#4535, #4536**,#4537,#4538,#4539));*

*#4549= IFCRELDEFINESBYPROPERTIES('1qWEOEXmTAQuOvpZxHLZTQ',#42,$,$,(#**4490**),#**4540**);*

Listing 5-6 Property information in IFC

This output can be also achieved by using the Revit Classification Manager and automatically populating both parameters, by assigning the IFC Class Mappings 4.1 classification, available by the tool.

5.2.4 Element Decomposition

Two façade types have been modeled in Revit – a façade with "Lamella" type panels and a mullion transom façade with glass panels (Figure 5-3).

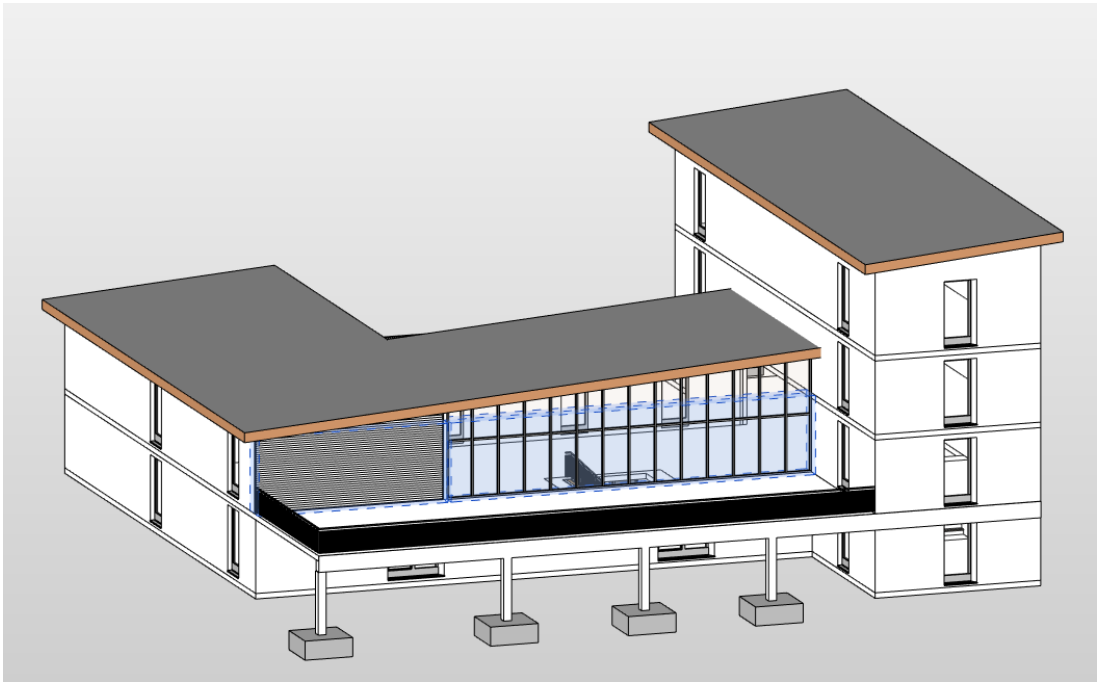

Figure 5-3 Facade elements in Revit

The decomposition of the curtain walls is represented by the concept of Element Decomposition. The relationship ifcRelAggregates contains both attributes – *RelatingObject* and *RelatedObjects*. The *RelatingObject*, here *IfcCurtainWall* (***#71368)*** is an object occurrence, that represents the assembled element and is the whole within the whole/part relationship. And the *RelatedObjects,* here *IfcPlate* and *IfcMember are the* objects, that are being aggregated.

#74415=
*IFCRELAGGREGATES('1tG6wy0Dr9AA4FvGWhSSOb',#42,$,$,**#71368**,(**#71428**,…,**#72407**,…));*

***#71368**= IFCCURTAINWALL('1tG6wy0Dr9AA4FuGWhSSOb',#42,'Curtain Wall:Lamellen-Einsatzelement:2452064',$,'Curtain Wall:Lamellen-Einsatzelement:613436',#71367,$,'2452064',$);*

***#71428**=*
IFCPLATE('1tG6wy0Dr9AA4FuGWhSSOa',#42,'Systemelement:Metallpanel:2452065',$,'Metallpanel', #71426,#71415,'2452065',.CURTAIN_PANEL.);

***#72407**= IFCMEMBER('1tG6wy0Dr9AA4FuGWhSSRg',#42,'Rechteckiger Pfosten:Lamellenrahmen:2452143',$,'Lamellenrahmen',#72402,#72396,'2452143',.MULLION.);*

Listing 5-7 Element decomposition in IFC

5.3 Validation

To test the use of the developed MVD, the IFC4 use case model has been validated against the mvdXML requirement sets and checked using the configurated Solibri rulesets. The focus of the validation through the generated mvdXML lies on checking for:

- Attribute existence in the *Applicability* element – check if a model element (e.g. *IfcWall*), is validated against the applicability constants (e.g. *PredefinedType='.* SOLIDWALL.'), exists in the IFC model.
- Attributes and properties existence – check if a model element contains attributes and properties of concept (e.g. Classification, Material, Property sets), that are required in a use case (BIM2Planon or BMA).
- Information location – check if information placed in the right location: e.g. the property *FireRating* exists in the property set *Pset_WallCommon*, applied to the entity *IfcWall.*
- Value existence, data type and valid syntax – check If properties are populated with values (e.g.Value[Exists]=TRUE), with a correct data type (e.g. Value[Type]='IfcLabel'); and check if constraints on data values are set, are they according to the predefined syntax (valid value for property *FireRating* is F90, whereas 'fire-resistant' is not).

Since the MVD specification does not include exchange requirements for the topics, listed below, the IFC model must be manually checked by architectural domain experts for their consistency and correctness:

- Correct design solutions and professional quality of the applied modeling techniques.
- If objects are modeled using the correct tools and/or if the correct object type is used – if façade is modeled and exported as *IfcWall* and not *IfcCurtainWall.*
- Relationships between objects – walls to slabs, or windows voiding a wall (a possible check with SMC rulesets).
- Clash detection (possible check with SMC rulesets).
- Existence and correctness of parameters, for which constraints have not been exclusively defined.

5.3.1 Model validation against Solibri rulesets

Solibri Model Checker (SMC) is a commercial checking platform, using hard-coded rules in high-level imperative programming languages such as C++ and Java, to perform clash detection and code compliance checking (Zhang et al. 2015). SMC provides predefined rule sets, created upon standardized model views and manuals, while user-defined rules can be configured to a certain extent in the Rule Manager tool. Until recently, the implementation of the customized rule sets was proprietary defined and limited to pre-defined parameters in these hard-wired rules. In October 2018 SCM announced new accessibility to ruleset creation and sharing (Solibri News 10/19/2018). To increase interoperability and customization freedom the platform launched a rule template API, allowing users to quickly generate and utilize self-created rules and rule sets. However, this feature is still in a closed beta phase and is not part of the scope of this work.

The configuration file, containing requirements on model elements, property sets, and properties, is generated by BIMQ in Excel format, according to building work phases and performing actors. The ruleset template configured by the Excel file is "Required property sets - SOL/203/2.4" (Figure 5-4). The rule checks if the model elements have the required property sets and properties, and if the values are correct.

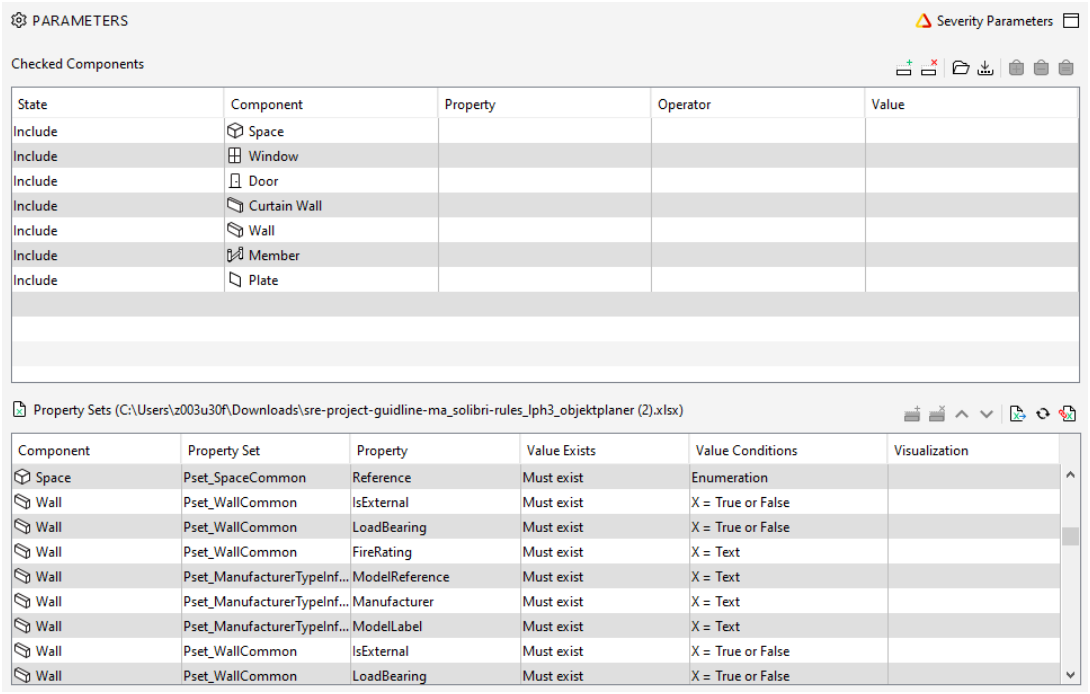

Figure 5-4 Configuration of checking rules in Solibri Ruleset Manager

Once the model check is performed, identified issues can be reported to the responsible party via BCF to execute required changes in the original BIM authoring software (Figure 5-5).

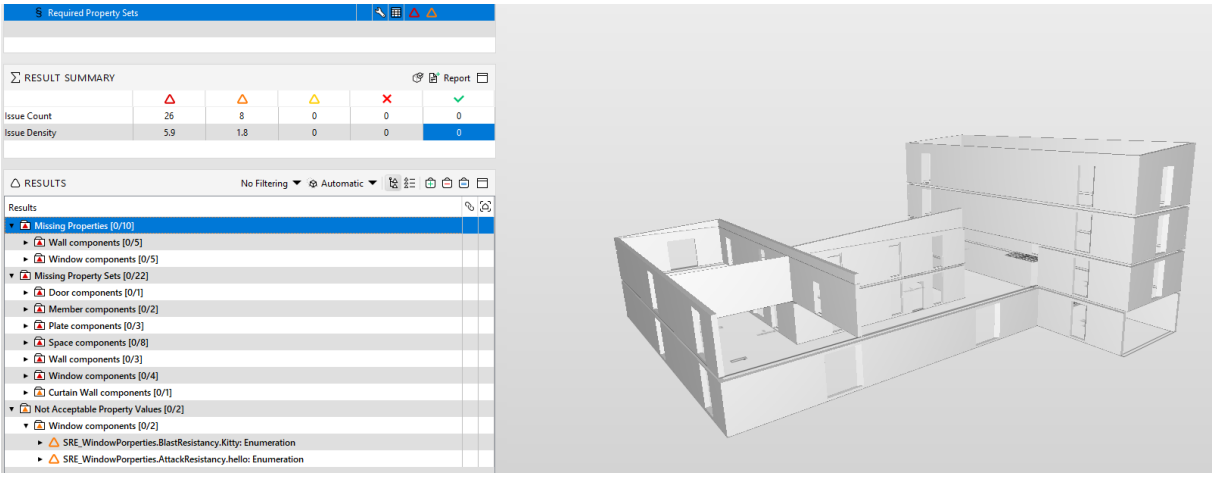

Figure 5-5 Validation checks in Solibri Model Checker

The examined workflow for validation of rules check with SMC has some limitations. For example, semantic information, such as window and door attributes, classification and material assignment, cannot be generated by BIMQ. Nevertheless, SMC offers some possibilities to incorporate similar checks by using different methods. To filter objects by specific criteria, similar to setting constraints in the *Applicability* element, an application-based Classification can be created (Figure 5-6). However, component properties can only reference information, contained in one of the five predefined groups, therefore checking if an object has a specific material concept assigned, according to the IFC format, would not be possible.

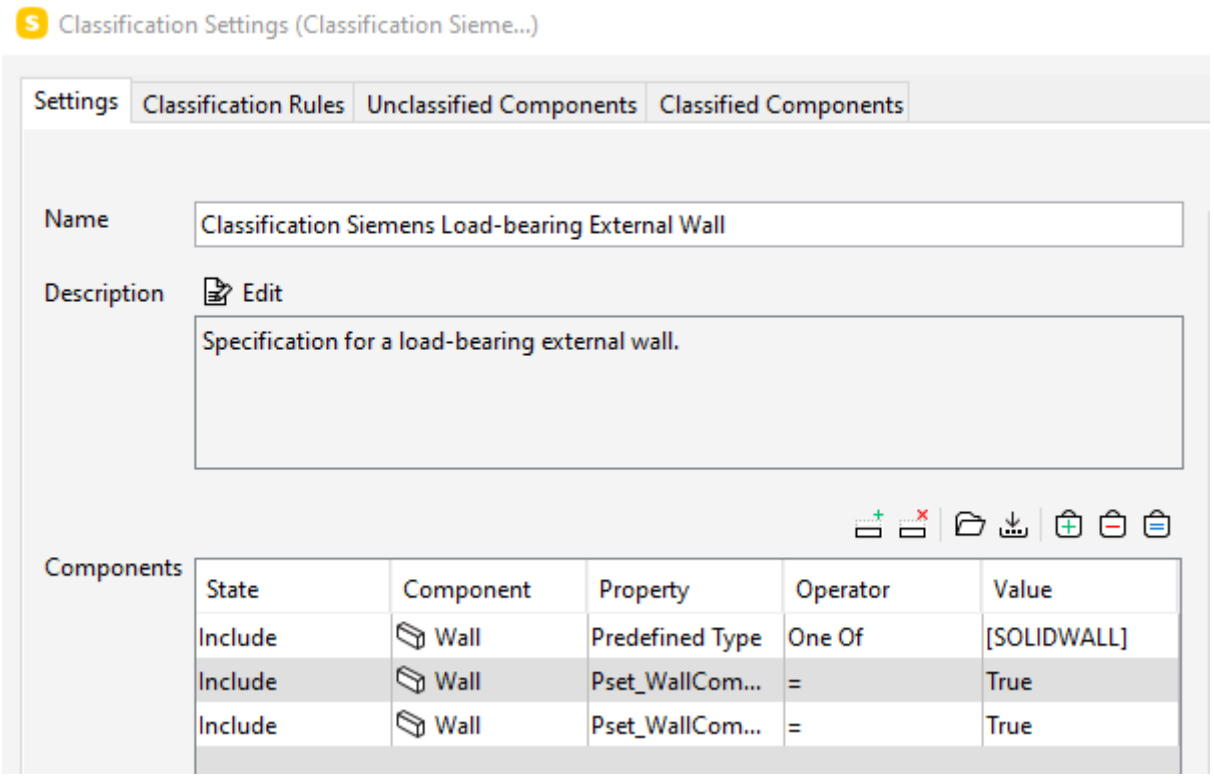

Figure 5-6 Classification settings in Solibri Model Checker

5.3.2 Model validation against mvdXML

To validate the IFC model against the generated mvdXML a few tools were considered. The choice of a suitable tool is limited by the two criteria– (1) validation of IFC4 data files and (2) import of mvdXML1.2 requirement sets. Not many tools on the market incorporate these features. Simplebim plugin, for example, was developed for the proprietary use of Statsbygg (STATSBYGG) and is not compatible with bSI's versions (Simplebim 2019).

The IfcDoc tool, for example, offers a direct MVD validation against an IFC file. Although all checks were marked as "passed", results show significant derivation from the actual data consistency in the IFC model, meaning that this feature does not present the actual validation results (Figure 5-7).

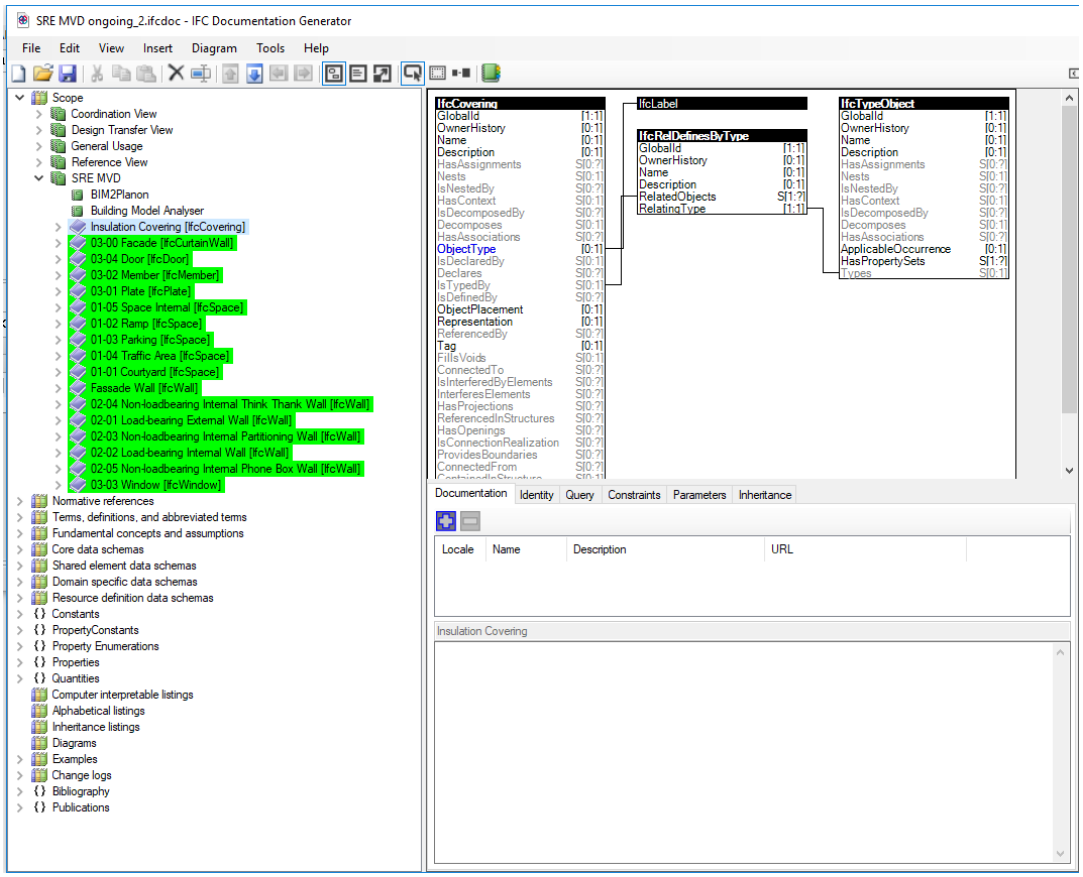

Figure 5-7 MVD validation in IfcDoc

xBim Xplorer is an application that can open, render and display data sematic from IFC2x3 and IFC4 models. Its open library allows the loading of plugins, that support IFC Schema validation and data extraction by syntax querying (XBIM n.d.). The tool supports validation against mvdXML 1.1. However, the IFC import was not fully successful and not all elements were checked (Figure 5-8).

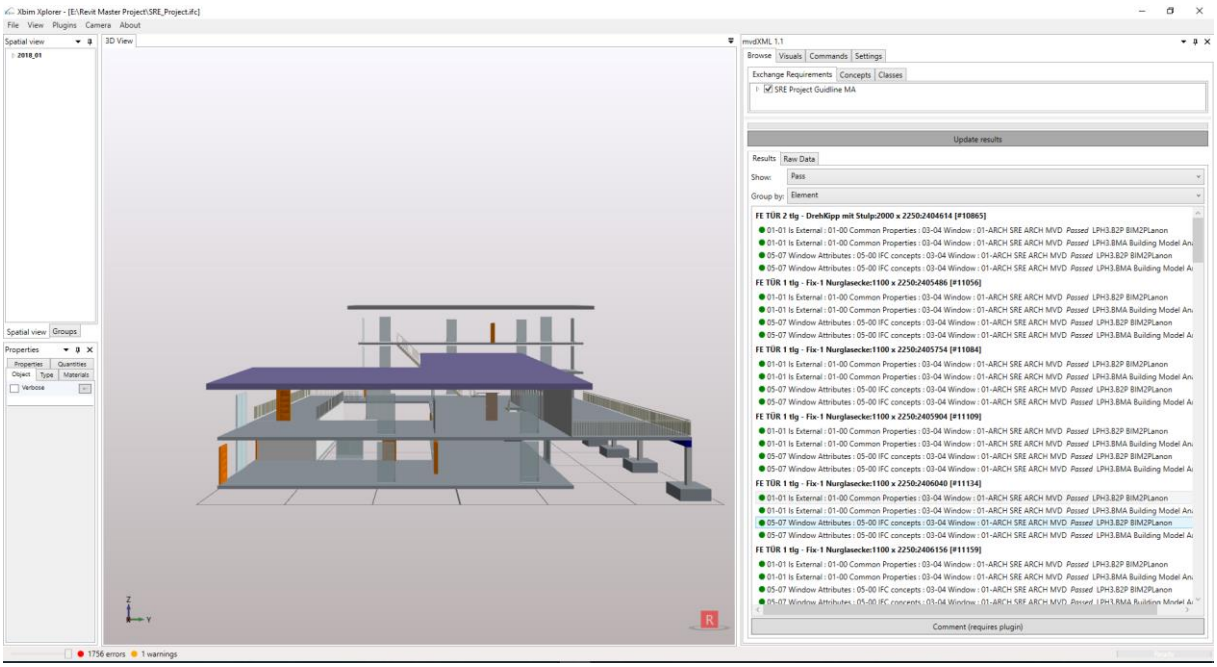

Figure 5-8 mvdXML validation in XbimXplorer

FZKViewer, developed by the Institute for Automation and Applied Informatics (Institute for Automation and Applied Informatics n.d.), is another software tool for visualization of semantic data models. FZKViewer Version 5.5 (still under development) supports mvdXML1.1 requirement sets and allows them to manipulate the mvdXML file. The user can select, edit and even create new rules, based on the existing mvdXML. Additionally, the IFC data model can be validated against a specific model view or an exchange requirement; *ConceptRoots*, *Concepts* and *RootEntities* can be individually selected and checked, providing flexibility and accuracy of the validation results (Figure 5-9).

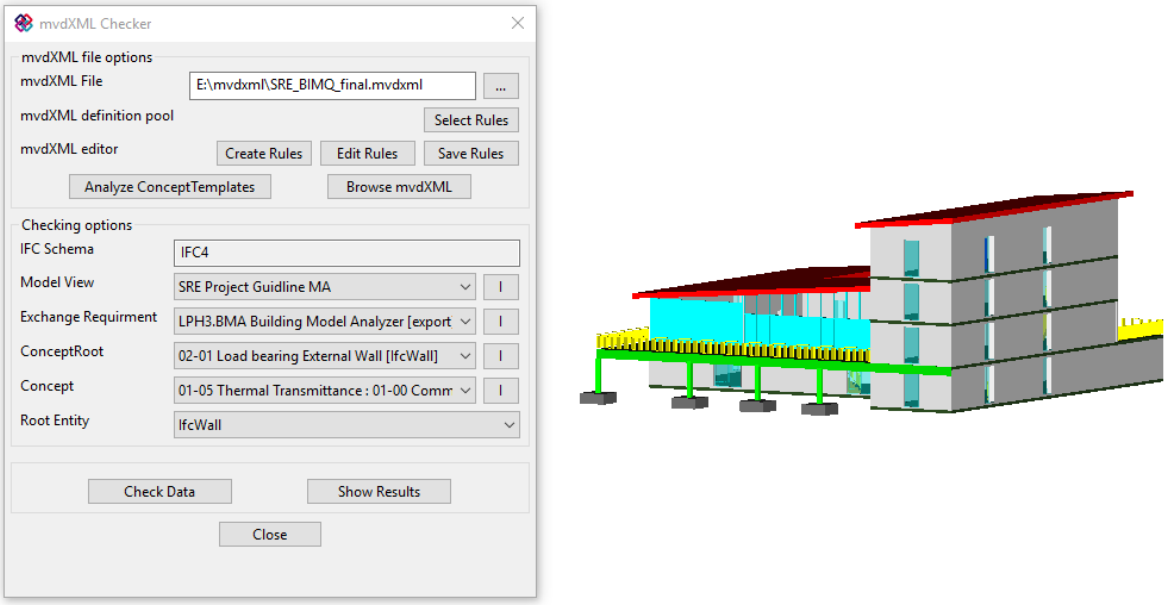

Figure 5-9 mvdXML checker setting options in FZKViewer

Figure 5-10 shows the detailed result report for the checks of *Pset_WallCommon* property set existence and IsExternal property value check on an *IfcWall* entity. This example shows that the property is provided to the individual wall object and not the object type, as well as, that the property values (*PropertyName[Value]='IsExternal'* AND *Value[Value]=TRUE*) are assigned correctly.

Successfull	Type: IfcWall | Name: Basic Wall:STB 250:2399834 | OID: #8233
Successfull	Check condition: PsetName [Value] == Pset_WallCommon (EXPRESS: IfcRoot.Name, Value: Pset_WallCommon)
Successfull	Check condition: PropertyName [Value] == IsExternal (EXPRESS: IfcProperty.Name, Value: [Reference|ExtendToStructure|IsExternal|LoadBearing|
Successfull	Check condition: Value [Value] == TRUE (EXPRESS: IfcPropertySingleValue.NominalValue, Value: [STB 250|TRUE|TRUE|TRUE|50|F60|4.184])
Failed	Check condition: TypePsetName [Value] == Pset_WallCommon (EXPRESS: IfcRoot.Name, Value:) - RuleIdState: AttributeNotFound
Failed	Check condition: TypePropertyName [Value] == IsExternal (EXPRESS: IfcProperty.Name, Value:) - RuleIdState: AttributeNotFound
Failed	Check condition: TypeValue [Value] == TRUE (EXPRESS: IfcPropertySingleValue.NominalValue, Value:) - RuleIdState: AttributeNotFound
Successfull	Check condition: PsetName [Value] == Pset_WallCommon (EXPRESS: IfcRoot.Name, Value: Pset_WallCommon)
Successfull	Check condition: PropertyName [Value] == IsExternal (EXPRESS: IfcProperty.Name, Value: [Reference|ExtendToStructure|IsExternal|LoadBearing|
Failed	Check condition: Value [Exists] == FALSE (EXPRESS: IfcPropertySingleValue.NominalValue, Value: [STB 250|TRUE|TRUE|TRUE|50|F60|4.184])

Figure 5-10 Results protocol in FZKViewer

5.4 Results and evaluation

The validation results and the deducted conclusions drawn are discussed below. The presented results are strongly depending on factors, such as the choice of validation tools, the used techniques, methods and approaches for data model creation, the MVD development itself and mvdXML format generation, therefore the outcomes are summarized respectively.

5.4.1 Software topics

The IFC data model is flexible when it comes to representing model data. This can be an obstacle when information has to be captured and validated against specific exchange requirements. Additionally, elements also differentiate in their geometrical representation, making it difficult to set specific requirements on how they are to be modeled and exported in an IFC. Therefore, project-specific requirements shall be always discussed with the designer, to clear such software tool details.

Currently, most of the available tools do not support error-free mvdXML1.1 validation, thus checking results are not completely reliable. Evaluation problems and verification errors can occur due to implementation problems, the inconsistency of the mvdXML or non-compliance with the XSD IFC schema. Additionally, once data quality issues are established by the validation, domain-specialist should be able to report and communicate the identified inconsistencies. The process of adding missing information or modifying model elements to conform to the rule checks can be partially realized by implementing a BCF exchange workflow. Issues are also caused by differences in how design software generate and export information. These problems can be solved by certifying IFC authoring tools for export and import of the latest IFC schema.

5.4.2 Mvdxml topics

The differences between the mvdXML, generated in IfcDoc (here shortly called IfcDoc mvdXML), and the mvdXML, configured in BIMQ (here shortly called BIMQ mvdXML), are discussed below.

- IfcDoc generates *ConceptTemplates*, that do not have a *Name* attribute, thus making the mvdXML not compatible with the IFC XSD schema.

```
    <ConceptTemplate uuid="00000000-0000-0000-0000-000000000000" name="_SRE
MVD" code="26d91a9d-b339-4a16-a8e5-b6c0472b2bbd" status="sample"
applicableSchema="IFC4">
      <SubTemplates>
        <ConceptTemplate uuid="a5708433-e7ca-45f4-8a6b-c8d1820cb746"
status="sample" applicableSchema="IFC4" applicableEntity="IfcSpace" />
        <ConceptTemplate uuid="d727af28-d205-4433-95d7-c2646ee5670d"
status="sample" applicableSchema="IFC4" applicableEntity="IfcDoor" />
        <ConceptTemplate uuid="690be133-f205-4ef5-8d29-f41af15a1471"
status="sample" applicableSchema="IFC4" applicableEntity="IfcCurtainWall"
/>
        <ConceptTemplate uuid="fdd068cb-7a1d-422f-811c-453e95a50e9d"
status="sample" applicableSchema="IFC4" applicableEntity="IfcWall" />
```

Listing 5-8 ConceptTemplates generated with no Name attribute in IfcDoc

- As observed by the MVD development in IfcDoc, currently it is not possible to reference more than one *ConceptTemplate* in the *Applicability* element. The template, used to generate the BIMQ mvdXML, allows a workaround by providing a *ConceptTemplate*, applicable entity *IfcProduct*. This custom created template includes *RuleIDs* for the attribute *PredefinedType* and *AttributeRules* for entities such as IfcRelDefinesByProperties, IfcRelDefinesByType and IfcRelAssociatesClassification. Theoretically, this allows to set more than one constraint and yet reference to only one *ConceptTemplate* (Listing 5-9).

```
        <ConceptRoot uuid="00e4509a-f99a-475d-b3eb-391688bfcb61" name="Load
bearing External Wall" status="sample" applicableRootEntity="IfcWall">
          <Definitions>
            <Definition>
              <Body><![CDATA[Specification for an external load-bearing
wall with Predfined Type SOLIDWALL.]]></Body>
            </Definition>
          </Definitions>
          <Applicability uuid="00000000-0000-0000-0000-000000000004"
status="sample">
            <Template ref="00000000-0000-0000-0001-000000000001" />
              <TemplateRules operator="and">
              <TemplateRule Parameters="PredefinedType[Value]='SOLIDWALL'
AND T_PredefinedType[Value]='SOLIDWALL'" />
              <TemplateRules operator="and">
              <TemplateRules operator="or">
              <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='LoadBearing' AND Value[Value]=TRUE" />
              <TemplateRules operator="and">
              <TemplateRule
Parameters="TypePsetName[Value]='Pset_WallCommon' AND
TypePropertyName[Value]='LoadBearing' AND TypeValue[Value]=TRUE" />
               <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='LoadBearing' AND Value[EXISTS]=FALSE" />
              </TemplateRules>
               </TemplateRules>
               </TemplateRules>
              <TemplateRules operator="and">
              <TemplateRules operator="or">
              <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='IsExternal' AND Value[Value]='TRUE" />
              <TemplateRules operator="and">
              <TemplateRule
Parameters="TypePsetName[Value]=''Pset_WallCommon' AND
TypePropertyName[Value]='IsExternal' AND TypeValue[Value]='TRUE" />
               <TemplateRule Parameters="PsetName[Value]='Pset_WallCommon'
AND PropertyName[Value]='IsExternal' AND Value[EXISTS]=FALSE" />
              </TemplateRules>
              </TemplateRules>
              </TemplateRules
              </TemplateRules>
          </Applicability>
```

Listing 5-9 Constraints in the Applicability element

- The mvdXML format allows extending the content of *TemplateRules* and the creation of rules on parameters, separated by logical operators. As shown in Listing 5-10 and Listing 5-11 the two concepts, that represent the same information vary in their syntax structure, depending on the used *ConceptTemplate.*

-

```
            <Concept uuid="00000301-3770-0000-0000-000000840867" name="05-
01 Classification : 05-00 IFC concepts : 01-05 Space Internal : 01-ARCH SRE
ARCH MVD" code="">
              <Definitions>
                <Definition>
                  <Body lang="en"><![CDATA[Classification : IFC concepts :
Space Internal]]></Body>
                </Definition>
              </Definitions>
              <Template ref="4a224609-6578-4c75-afcf-8affa86e5ef2"/>
              <Requirements>
                <Requirement applicability="export" exchangeRequire-
ment="00000301-3770-3770-0000-000000000000" requirement="mandatory"/>
                <Requirement applicability="export" exchangeRequire-
ment="00000301-3770-3769-0000-000000000000" requirement="mandatory"/>
              </Requirements>
         <TemplateRules operator="and">
              <TemplateRule Parameters="Name[Exists]='TRUE'"/>
         <TemplateRule Parameters="Identification[Exists]='TRUE'"/>
         <TemplateRule Parameters="ClassificationName[Value]='UniClass'"/>
        <TemplateRule Parameters="ClassificationName[Value]='CAFM'"/>
               </TemplateRules>
           </Concept>
```

Listing 5-10 XML syntax in the mvdXML, generated by BIMQ

```
            <Concept uuid="93f242f3-2daf-4d4b-9dc7-f8c0ecc743c8"
name="Classification" status="sample" override="false">
              <Template ref="4a224609-6578-4c75-afcf-8affa86e5ef2" />
              <Requirements>
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="7fab58cb-4655-489a-862d-85e26a3096b2" />
                <Requirement applicability="export" requirement="mandatory"
exchangeRequirement="cfa72802-9d70-407c-9616-56f5b53f8708" />
              </Requirements>
              <TemplateRules operator="and">
     <TemplateRule xsi:type="TemplateItem" Parameters="Classification-
Name[Value]='CAFM'">
                  <References />
                </TemplateRule>
     <TemplateRule xsi:type="TemplateItem" Parameters="Classification-
Name[Value]='Uniclass'">
                  <References />
                </TemplateRule>
     <TemplateRule xsi:type="TemplateItem" Parameters="Name[Exists]=TRUE">
                  <References />
                </TemplateRule>
     <TemplateRule xsi:type="TemplateItem"  Parameters="Identification[Ex-
ists]=TRUE">
           <References />
           </TemplateRule>
         </TemplateRules>
       </Concept>
     </Concepts>
   </ConceptRoot>
```

Listing 5-11 XML syntax in the mvdXML, generated by IfcDoc

5.5 Conclusion

The presented case study shows, that validating an IFC data model against the developed mvdXML-format based MVD is essential for data quality, when transferring and maintaining information between different tools. A fully automated data analysis process requires comprehensive definitions and specifications of exchange requirements, that will also minimize design mistakes and miscommunication, due to lack of information and certainty. Nevertheless, design software that exports IFC must offer the tools and features to create content, aligned with specified requirements. Validation results also lead to the conclusion that further development of mvdXML generating and - validating tools is required. Since generation and validation of MVD in IfcDoc is not based on the mvdXML format, incorrect validations, because of incompatibility or syntax mistakes, are minimized. Therefore, the further development of the tool is crucial.

6 Discussion

This chapter provides a summary of the examined work and results. Furthermore, based on the observed implementation problems in a praxis-related example, suggestions for future work and topics of development are discussed.

6.1 Summary

In summary, the outcome of this work has met the aimed goal in terms of outlining use cases in the early building design for cost simulation (BMA) and facility management documentation (BIM2Planon), and translating the BIM@SRE guidelines into a machine-readable format. Additionally, exchange requirements were defined to quantitatively depict the term “model quality”; a Model View Definition was developed to specifically indicate data content and format, and different tools (IfcDoc and BIMQ) were used to configure the MVD and generate an mvdXML. An IFC data model was created and validated against mvdXML. Limitations have been outlined at each step of the MVD development and suggested alternative workflows have been discussed. Examined validation results show, that information management is the key to a successful implementation of BIM, which can be achieved by using open specifications with clear definitions of terms, concepts and information requirements.

6.2 Future work

Based on the evaluated results and observed limitations, suggestions on which approaches should be pursued by different stakeholders and which areas of work need further investigations, are discussed.

6.2.1 Stakeholder outlook

Based on the outcomes of this work, suggestions for further development in the following aspects, relevant for the building owner Siemens, are presented:

- Develop requirement specifications of objects’ geometrical detail level (LoG) to allow data validation against other internal guidelines and examine further topics such as fire code compliance checking, structural analysis and others.
- Develop Structural Engineering and MEP exchange requirements templates, since they are a major cost-driven factor at later project development.

- Consider the level of fuzziness in the early design phase and implement not only consistency checks but also consider and examine the information uncertainty percentage of properties and parameters.
- Consider requirement sets for disciplines such as fire and acoustic engineering that do not require modeling specific objects in the model, but set requirements for other discipline models, e.g. architectural models.
- Outline further use cases of interest to focus on topics such as life cycle assessment, operation costs and facility maintenance, energy consumption and environmental impact.
- Run pilot projects and consult with domain experts to test and improve MVDs content.
- Emphasize the need for further development of tools such as IfcDoc and BIMQ to match modern expectations and include all functionalities needed to enable a cohesive and effective digital requirement transfer.

6.2.2 MvdXML improvements

The lack of standardization in terms of mvdXML development provides a high degree of flexibility when creating custom *ConceptTemplates.* This results in the existence of multiple mvdXML, that serve similar or even identical use cases, but differentiate themselves widely in their structure and syntax grammar. One of the possible solutions is for bSI to provide mvdXML base documents, which define all needed *ConceptTemplates*. This approach will limit the users' development freedom to mainly setting rules with parameters and validating data, but will also allow software vendors to specialize in import and export of the unified *ConceptTemplates.*

The bSI Model Support Group has presented improvement suggestions on the mvdXML specification at the buildingSMART Summit 2019 (Matthias Weise 2019). The suggested topics focus mainly on IFC data validation with mvdXML and include features like

- Units checks
- Checks for the existence of objects
- Nesting concepts, that allow defining ERs for individual instances
- Use more than one ConceptTemplate in a checking rule
- Conditional statements

As shown in the scope of this work, developing and configuring a complex MVD, that fully comprises with the defined exchange requirements, cannot be efficiently used in the praxis, if the tools that author and validate the data, do not cover the specter of features, needed for the specified exchange requirements. Nevertheless, such MVDs could be used for generating an IFC file that complies with a specific quality standard and guarantees building information consistency. This beneficial scenario has been acknowledged by stakeholders and the demand for standardization and specifications arises.

References

AEC3 (2019): BIMQ Platform. Available online at https://bim-plattform.com/en/bimq/.

Autodesk Inc: Improving Building Design Project Collaboration using openBIM® Data Exchange Standards. https://damassets.autodesk.net/content/dam/autodesk/www/campaigns/interoperability/fy17-aec-ifc-interoperability-whitepaper-en.pdf, accessed on 07.01.2020.

Autodesk Inc (2018): Revit IFC manual. Detailed instructons for handling IFC files.

Baldwin, Mark (2017): Der BIM-Manager. Praktische Anleitung für das BIM-Projektmanagement. [s.l.]: Beuth (Beuth Innovation).

Baumgärtel, Ken; Pirnbaum, Stephan; Pruvost, Hervé; Scherer, Raimar (2016): Automatic BIM filtering using Model View Definitions.

DIN ISO 29481-1: Bauwerks-Informations-Modelle – Informations-Lieferungs-Handbuch – Teil 1: Methodik und Format.

BIM Loket (2019): BIM basic IDM. Available online at https://www.bimloket.nl/BIMbasicIDM.

BIMQ (2018): Information Management with BIMQ. Webinar Jun 21, 2018. Available online at https://www.youtube.com/watch?v=amrW4qrhM4A&t=436s, accessed on 07.01.2020.

BIMtech: BIMtech. Available online at http://bimtech.io/site/.

BKI (2019): Baukosten Gebäude + Bauelemente + Positionen Neubau 2019. Statistische Kostenkennwerte Teil 1 + Teil 2 + Teil 3. Stuttgart: BKI.

BLIS Consortium - Digital Alchemy (2019): IFC Solutions Factory – The Model View Definition. Available online at http://www.blis-project.org/IAI-MVD/reporting/listMVDs_4.php?SRT=Name&MVD=. , accessed on 07.01.2020.

BLIS Project: BLIS Coordiation Project. Available online at https://www.blis-project.org/index2.html. , accessed on 07.01.2020.

Borrmann, André; König, Markus; Koch, Christian; Beetz, Jakob (2018): Building information modeling. Technology foundations and industry practice / André Borrmann, Markus Konig, Christian Koch, Jakob Beetz, editors. Cham, Switzerland: Springer.

buildingSMART International: Model View Definition (MVD) - An Introduction. Available online at https://technical.buildingsmart.org/standards/mvd/. , accessed on 07.01.2020.

buildingSMART International (2019a): buildingSMART Data Dictionary (bSDD). Available online at https://technical.buildingsmart.org/standards/bsdd/. , accessed on 07.01.2020.

buildingSMART International (2019b): buildingSMART Data Dictionary (bSDD). Available online at https://technical.buildingsmart.org/standards/bsdd/. , accessed on 07.01.2020.

buildingSMART International (2019c): IFC4_ADD2_TC1 - 4.0.2.1. IFC4 Documentation. IfcMaterial. Available online at https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/schema/ifcmaterialresource/lexical/ifcmaterial.htm. , accessed on 07.01.2020.

buildingSMART International (2019d): IFC4_ADD2_TC1 - 4.0.2.1. IFC4 Documentation. Classification. Available online at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/templates/classification.htm. , accessed on 07.01.2020.

buildingSMART International (2019e): IFC4_ADD2_TC1 - 4.0.2.1. Introduction. Available online at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/introduction.htm. , accessed on 07.01.2020.

buildingSMART International (2019f): IFC4_ADD2_TC1 - 4.0.2.1. Object Occurrence Attributes. Available online at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/. , accessed on 07.01.2020.

buildingSMART International (2019g): IFC4_ADD2_TC1 - 4.0.2.1. Property Sets for Objects. Available online at https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/schema/templates/property-sets-for-objects.htm. , accessed on 07.01.2020.

buildingSMART International (2019h): IfcDoc. Available online at https://technical.buildingsmart.org/resources/ifcdoc/. , accessed on 07.01.2020.

buildingSMART International (2019i): Industry Foundation Classes (IFC) - An Introduction. Available online at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/. , accessed on 07.01.2020.

buildingSMART International (2019j): Material Association. Available online at https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/templates/material-association.htm. , accessed on 07.01.2020.

buildingSMART International (2019k): MVD Database. Available online at https://technical.buildingsmart.org/standards/mvd/mvd-database/. , accessed on 07.01.2020.

buildingSMART International (2019l): mvdXML. Available online at https://technical.buildingsmart.org/standards/mvd/mvdxml/. , accessed on 07.01.2020.

buildingSMART International, BLIS Consortium/Richard See (2012): An Integrated Process for Delivering IFC Based Data Exchange. Available online at https://standards.buildingsmart.org/documents/IDM/IDM_guide-IntegratedProcess-2012_09.pdf.

Bundesministerium fur Verkehr und digitale Infrastruktur (2018): Stufenplan Digitales Planen und Bauen. Technischer Bericht. Available online at https://www.vbw-bayern.de/Redaktion/Frei-zugaengliche-Medien/Abteilungen-GS/Planung-und-Koordination/2018/Downloads/Studie-Digitales-Planen-und-Bauen.pdf.

CAFM-Connect: CAFM-Connect Complete catalogue. CC3.0 - EN. Available online at https://bim-profile.cafm-connect.org/ProfileViewer.aspx?profile=CC3.0%20-%20EN&version=1.0.

Chipman, Tim; Liebich, Thomas; Weise, Matthias (2016): mvdXML specification 1.1. Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules.

Christian Schittlich (Ed.) (2012): im Detail Eibfach Bauen Zwei. Available online at https://issuu.com/detail-magazine/docs/978-3-920034-62-1-bk-de_einfachbaue.

Die Deutsche Bauindustrie (2019): BIM im Hochbau. Technisches Positionspapier. Available online at https://www.bauindustrie.de/presse/presseinformationen/positionspapier-bim-im-hochbau/.

DIN EN ISO 16739, April 2017: DIN EN ISO 16739.

Dr Stephen Hamil (2018): What is COBie? Available online at https://www.thenbs.com/knowledge/what-is-cobie.

Emma Hooper: IFC (Industry Foundation Classes) – Layers and Classification in Autodesk Revit. Bond Bryan Digital. Available online at https://bimblog.bondbryan.co.uk/ifc-industry-foundation-classes-layers-and-classification-in-autodesk-revit/.

Global Construction Perspectives and Oxford Economics (2015): Global Construction 2030: A global forecast for the construction industry to 2030. Available online at https://www.pwc.se/sv/entreprenad/assets/global-construction-2030.pdf.

Institute for Automation and Applied Informatics (n.d.): FZKViewer. https://www.iai.kit.edu/english/1648.php.

J. Abualdenien; A. Borrmann (2019): A meta-model approach for formal specification and A meta-model approach for formal specification and consistent management of multi-LOD building models. Advanced Engineering Informatics. Available online at https://publications.cms.bgu.tum.de/2019_abualdenien_multi-LOD_consistency.pdf.

Jiang, Shaohua; Wu, Zheng; Zhang, Bo; Cha, Hee (2019): Combined MvdXML and Semantic Technologies for Green Construction Code Checking. In *Applied Sciences* 9, p. 1463. DOI: 10.3390/app9071463.

Jimmy Abualdenien; Sina Pfuhl; Alexander Braun (2019): Development of an MVD for checking fire-safety and pedestrian simulation requirements. 1Chair of Computational Modeling and Simulation, Technical University of Munich. 31. Forum Bauinformatik, Berlin, 2019. Available online at https://publications.cms.bgu.tum.de/abualdenien_pfuhl_braun_fbi2019.pdf.

Lee, Yongcheol; Eastman, Charles; Solihin, Wawan (2016): An ontology-based approach for developing data exchange requirements and model views of building information modeling. In *Advanced Engineering Informatics* 30, pp. 354–367. DOI: 10.1016/j.aei.2016.04.008.

Matthias Weise (2019): IFC Validation Checks with mvdXML2. Status and future directions of mvdXML. buildingSMART Summit 2019. Available online at https://forums.buildingsmart.org/t/mvdxml-improvements/2117.

McKinsey Productivity Sciences Center, Singapore (2016): Imagining-constructions-digital-future.

Pinheiro, Sergio; Corry, Edward; Kenny, Paul; O Donnell, James (2015): Development of a Model View Definition for Environmental and Energy Performance Assessment.

Regimantas Ramanauskas (2018): Managing and Analysing BIM Data With Revit + Dynamo + Power BI. LinkedIn. Available online at https://www.linkedin.com/pulse/managing-analysing-bim-data-revit-dynamo-power-bi-ramanauskas/.

Richard Kelly (2019): Digital transformation to improve the built asset industry. buildingSMART Summit, Beijing, 28 October 2019. LinkedIn. Available online at https://www.linkedin.com/pulse/solutions-standards-program-buildingsmart-summit-beijing-kelly/.

Rijksgebouwendienst (2012): Rgd BIM Standard. Available online at https://english.rijksvastgoedbedrijf.nl/binaries/central-government-real-estate-agency/documents/publication/2014/07/08/rgd-bim-standard-v1.0.1-en-v1.0_2/Rgd_BIM_Standard_v1_0_1_EN_v1_0__2_.pdf.

Robert Amor (2019): Great Progress in Code Compliance Checking. Leonhard Obermayer Center day, 11/15/2019.

Roland Berger (2017): Turning point for the construction industry. The disruptive impact of Building Information Modeling (BIM).

Sacks, Rafael; Kedar, Amir; Borrmann, Andre; Ma, Ling; Brilakis, Ioannis; Hüthwohl, Philipp et al. (2018): SeeBridge as next generation bridge inspection: Overview, Information Delivery Manual and Model View Definition. In *Automation in Construction* 90, pp. 134–145. DOI: 10.1016/j.autcon.2018.02.033.

Siemens AG Real Estate (2017): BIM@SRE. BIM@Siemens Real Estate (Version 2.0). Available online at https://assets.new.siemens.com/siemens/assets/api/uuid:d90d97488498e68e7332a6872cff10a29fc33f03/version:1520000436/bim-standard-siemens-real-estate-version-2-0-de.pdf.

Simplebim (2019): mvdXML add-on for Simplebim 8.0. Available online at http://datacubist.com/support/addon-mvdxml.html.

Sina Pfuhl (October/2018a): Analysis of Exchange Requirements for BIMbased Fire Code Compliance Checking. Bachelorthesis. TUM Department of Civil, Geo and Environmental Engineering.

Sina Pfuhl (October 2018b): Analysis of Exchange Requirements for BIMbased Fire Code Compliance Checking. Bachelorthesis. TUM Department of Civil, Geo and Environmental Engineering.

Solibri News (10/19/2018): Boosting quality through unbeatable interoperability and customization. Helsinki, Finland. Available online at https://www.solibri.com/news/boosting-quality-through-unbeatable-interoperability-and-customization.

Solihin, Wawan; Eastman, Charles; Lee, Yong-Cheol (2015): Toward robust and quantifiable automated IFC quality validation. In *Advanced Engineering Informatics* 29 (3), pp. 739–756. DOI: 10.1016/J.AEI.2015.07.006.

STATSBYGG: SIMBA - Statsbygg's BIM. Available online at https://sites.google.com/view/simba-bim-krav.

STATSBYGG (2019): STATSBYGG BIM-MANUAL 2.0 (SBM2). Available online at http://www.eubim.eu/wp-content/uploads/2019/08/2019-08-28_EU_BIM_Task_Group_Statsbygg_BIM_Manual_20_v101.pdf.

Tony Cunningham (2013): Factors Affecting The Cost of Building Work - An Overview. Dublin Institute of Technology. Available online at https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1028&context=beschreoth.

Windisch, Ronny; Katranuschkov, Peter; Scherer, R. (2012): A Generic Filter Framework for Consistent Generation of BIM-based Model Views.

XBIM (n.d.): xBIMXplorer. Available online at https://docs.xbim.net/downloads/xbimxplorer.html.

Yalcinkaya, Mehmet; Singh, Vishal (2019): VisualCOBie for facilities management. In *Facilities* 37 (7/8), pp. 502–524. DOI: 10.1108/F-01-2018-0011.

Zhang, Chi; Beetz, Jakob (2014): Model View Checking: Automated Validation for IFC Building Models.

Zhang, Chi; Beetz, Jakob; Vries, de (2013): Towards model view definition on semantic level : a state of the art review.

Zhang, Chi; Beetz, Jakob; Weise, Matthias (2015): Interoperable validation for IFC building models using open standards. In *Electronic Journal of Information Technology in Construction* 20, pp. 24–39.

Attachment A

Attachment B

The enclosed CD contains the following content:

- The written part of the work as a Word document
- The created IFC model and the associated Revit project
- The source code of the developed applications

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-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, 14. January 2020

Vorname Nachname

Gergana Popgavrilova

[redacted]