



# **BIM-Based Code Compliance Checking of the Musterbauordnung**

Scientific work to obtain the degree

**Master of Science (M.Sc.)**

at the Department of Civil, Geo and Environmental Engineering of the  
Technical University of Munich.

**Supervised by** Prof. Dr.-Ing. André Borrmann  
Simon Vilgertshofer, M.Sc.  
Lehrstuhl für Computergestützte Modellierung und Simulation

**Submitted by** Sebastian Schliski  
██████████

**Submitted on** July 15, 2021



## Abstract

The present work deals with the Automated Code Compliance Checking (ACCC) of 3D building models and outlines a methodology for translating content from standards and legal texts into machine-interpretable information, thus enabling an automated verification of these contents. Therefore, the fifth section of the Model Building Regulation (Musterbauordnung) (MBO) is the central point of the investigations.

The digitization of the Architecture, Engineering and Construction (AEC) sector is steadily progressing and is significantly influenced by the Building Information Modeling (BIM) method. BIM utilizes digital building models instead of 2D plans for a building's planning, construction, and maintenance throughout its entire life cycle. All project participants incorporate their planning services as model content into the building model. This change in the way of working aims at reducing planning errors and the resulting costs and delays. In order to assure high model quality, consistent quality and conformity checking of all model contents have to be conducted.

Design results of engineers and architects are claimed and regulated by standards and legal texts. The underlying models and model contents reflect the quality of this planning performance. In order to be able to carry out an automated check concerning these regulations, a translation of the demanded contents into machine-interpretable information is required. The development of code compliance checking tests is complicated because of the very high complexity of the respective standards or legal texts.

For this purpose, this work outlines a methodology for identifying and processing relevant data. To assess the absence or presence of information, detailed knowledge of the used data format is required. For this purpose, appropriate classes of the open data format Industry Foundation Classes (IFC) are discussed regarding automated compliance checks, and essential information for testing the fifth section of the MBO are identified. Therefore, an approach is developed to check the correctness of the resulting mapping process into the IFC format and adherence to outlined exchange requirements.

In conclusion, the described methodology for enabling ACCC of regulatory requirements is tested for its stability based on selected sections of the MBO and a sample building model.

## Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der automatisierten Konformitätsprüfung von 3D Gebäudemodellen und legt dabei eine Methodik zugrunde, Inhalte aus Normen und Gesetzestexten in maschinen-interpretierbare Informationen zu übersetzen und somit eine automatisierte Prüfung dieser Inhalte zu ermöglichen. Dabei stellt der fünfte Abschnitt der Musterbauordnung den zentralen Punkt der Untersuchungen dar.

Die Digitalisierung der Baubranche schreitet stetig voran und wird dabei maßgeblich durch die Methode des Building Information Modeling (BIM) geprägt. Hierbei werden digitale Gebäudemodelle verwendet, um eine effizientere Planung und Erhaltung von Gebäuden zu ermöglichen. Alle Projektbeteiligten lassen ihre Planungsleistung als Modellinhalte in das Gebäudemodell miteinfließen. Diese Veränderung der Arbeitsweise verfolgt das Ziel der Reduzierung von Planungsfehler und der daraus entstehenden Kosten bzw. Verzögerungen. Um dieses Ziel gewährleisten zu können, bedarf es konsequenter Qualitäts- und Konformitätsprüfung aller Modellinhalte.

Planungsergebnisse von Ingenieuren und Architekten werden durch Normen und Gesetzestexte gefordert und reguliert. Die Qualität dieser planerischen Leistung spiegelt sich in den zugrundeliegenden Modellen bzw. Modellinhalten wider. Um ein Modell hinsichtlich dieser Anforderungen automatisiert prüfen zu können, wird eine Übersetzung der geforderten Inhalte in maschinen-interpretierbare Informationen benötigt. Die Entwicklung solcher Konformitätsprüfungen ist durch die meist hohe Komplexität der jeweiligen Normen bzw. Gesetzestexte sehr kompliziert.

Deshalb wird in der vorliegenden Arbeit ein Konzept zur Identifikation und Aufbereitung relevanter Daten dargelegt. Um das Fehlen bzw. Vorhandensein von Informationen beurteilen zu können, wird ein detailliertes Wissen hinsichtlich des verwendeten Datenformates benötigt. Hierzu werden für die automatisierte Prüfung relevante Klassen aus dem offenen Datenformat IFC diskutiert und zusätzliche essenzielle Informationen zur Prüfung der Musterbauordnung identifiziert, die per Attribut in das Modell übergeben werden. Desweiteren wird ein Ansatz zur anschließenden Weiterverarbeitung der Daten und des daraus resultierende Übertragungsprozesses in das Datenformat IFC offengelegt. Dazu wird der Übertragungsprozess der zusätzlich nötigen Informationen auf seine Richtigkeit bzw. Vollständigkeit überprüft und die Einhaltung der dargelegten Austauschforderungen verifiziert.

Anschließend wird das dargelegte Konzept zur automatisierten Prüfung anhand ausgewählter Ausschnitte der MBO und eines Probegebäudemodells auf ihre Tragfähigkeit geprüft und deren Ergebnisse diskutiert.



# Contents

<b>Contents</b>	<b>V</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Scope . . . . .	2
1.3 Structure of work . . . . .	2
<b>2 Introduction to IFC, IDM, MVD and Co.</b>	<b>4</b>
2.1 Historical Background . . . . .	4
2.2 Industry Foundation Class . . . . .	4
2.2.1 IFC Layers . . . . .	5
2.2.2 Inheritance Hierarchy . . . . .	7
2.2.3 Object Relationship . . . . .	8
2.2.4 Technological Outlook . . . . .	10
2.3 Information Delivery Manual and Model View Definitions . . . . .	10
2.3.1 mvdXML . . . . .	12
2.4 LOD, LOG, LOI . . . . .	14
2.5 Summary . . . . .	15
<b>3 Code Compliance Checking</b>	<b>16</b>
3.1 Introduction . . . . .	16
3.2 Levels of Model Quality . . . . .	17
3.3 Technical Approaches . . . . .	18
3.3.1 Hard-Coded Tests . . . . .	18
3.3.2 Domain Specific Programming Languages . . . . .	21

3.3.3	Visual Programming Language . . . . .	22
3.4	Summary . . . . .	25
<b>4</b>	<b>Musterbauordnung</b>	<b>27</b>
4.1	Introduction . . . . .	27
4.2	Fifth Section of Musterbauordnung . . . . .	27
4.3	Representation & Complexity of Regulatory Requirements . . . . .	29
4.3.1	Representation . . . . .	29
4.3.2	Complexity . . . . .	30
4.4	Extraction Methods . . . . .	31
4.4.1	Semantic Mark-Up RASE Methodology . . . . .	31
4.4.2	Natural Language Processing . . . . .	33
4.5	Summary . . . . .	36
<b>5</b>	<b>Concept</b>	<b>37</b>
5.1	RASE Mark Up Technique . . . . .	37
5.2	IFC Data Coverage . . . . .	39
5.2.1	Three-Dimensional Aggregation . . . . .	39
5.2.2	Building Element Aggregation . . . . .	43
5.2.3	Additional Semantic Assignment . . . . .	46
5.2.4	Building Element Allocation . . . . .	47
5.3	Data Extraction and Mapping . . . . .	49
5.3.1	Granularity and Complexity . . . . .	50
5.3.2	Further Analysis and Strategies . . . . .	52
5.3.3	Data Mapping Tables . . . . .	54
5.3.4	Code Compliance Checking Approaches . . . . .	58
5.4	Technical Implementation . . . . .	59

5.5	Summary	62
<b>6</b>	<b>Proof of Concept</b>	<b>64</b>
6.1	Additional Data Identification	64
6.2	Model Data Quality	67
6.3	Use Cases	70
6.3.1	§33 - First and Second Escape Way	70
6.3.2	§ 34 - Stairs	74
6.4	Validation	77
6.4.1	Validation of Data Quality	77
6.4.2	Validation of Design Quality	79
6.5	Approach Evaluation and Limitations	83
6.6	Summary	85
<b>7</b>	<b>Conclusion</b>	<b>86</b>
7.1	Summary	86
7.2	Future Work	88
<b>A</b>		<b>90</b>
	<b>List of Figures</b>	<b>91</b>
	<b>List of Tables</b>	<b>94</b>
	<b>List of Algorithms</b>	<b>95</b>
	<b>References</b>	<b>96</b>





# Acronyms

<b>ACCC</b>	Automated Code Compliance Checking
<b>AEC</b>	Architecture, Engineering and Construction
<b>AI</b>	Artificial Intelligence
<b>API</b>	Application Programming Interface
<b>BCA</b>	Building Construction Authority
<b>BCF</b>	BIM Collaboration Format
<b>BEP</b>	BIM Execution Plan
<b>BERA</b>	Building Environment Rule and Analysis
<b>BIM</b>	Building Information Modeling
<b>BOM</b>	BERA Object Model
<b>BPMN</b>	Business Process Modeling Notation
<b>bsDD</b>	buildingSMART Data Dictionary
<b>bSI</b>	buildingSMART International
<b>CDE</b>	Common Data Environment
<b>CFG</b>	Context-Free Grammar
<b>CORENET</b>	Construction and Real Estate Network
<b>EIR</b>	Exchange Information Requirements
<b>ER</b>	Exchange Requirements
<b>GUI</b>	Graphical User Interface
<b>HOAI</b>	Fee Structure for Architects and Engineers
<b>IAI</b>	International Alliance for Interoperability
<b>IDM</b>	Information Delivery Manual
<b>IDS</b>	Information Delivery Specification
<b>IFC</b>	Industry Foundation Classes
<b>ISO</b>	International Organization for Standardization
<b>LoD</b>	Level of Detail
<b>LOD</b>	Level of Development
<b>LoG</b>	Level of Geometry
<b>LoI</b>	Level of Information
<b>MBO</b>	Model Building Regulation (Musterbauordnung)
<b>ML</b>	Machine Learning
<b>MVD</b>	Model View Definition
<b>NLP</b>	Natural Language Processing
<b>OWL</b>	Ontology Web Language
<b>POS</b>	Part-Of-Speech
<b>PSG</b>	Phrase Structure Grammar
<b>RDF</b>	Resource Description Language
<b>STEP</b>	Standard for the Exchange of Product Model Data

<b>UID</b>	Unique Identifier
<b>VCCL</b>	Visual Code Checking Language
<b>VPL</b>	Visual Programming Language
<b>XML</b>	Extensible Markup Language

# Chapter 1

## Introduction

### 1.1 Motivation

The last century was affected by high demands for living and working spaces in metropolitan areas, resulting in rising rent and purchase prices worldwide. These issues concern social and economic parts of life and increase the pressure on society, companies, and governments. Additionally, more than a third of global CO<sub>2</sub> emissions are caused by the building and construction sector if upstream power generation is taken into consideration (JONES and HOWARTH, 2019). Therefore, inaccurate planning and construction affect the issues mentioned above. Due to this, the Architecture, Engineering, and Construction (AEC) sector holds a special responsibility in terms of enhancing current planning, production, and sustainability processes.

In order to meet these demands, Building Information Modeling (BIM) helps to reduce time and cost-consuming errors in the planning and production phases. Roland Berger states the substantial impacts of digital technologies, like BIM, on business models and emphasizes the potential in improving the overall efficiency (“Roland Berger Focus”, 2017).

Nevertheless, companies of the AEC sector need to invest more money in digitization. According to McKinsey Global Institute, the construction sector is the second-lowest digitized industry in 2015 and was not able to catch up in last years (“McKinsey”, 2016, RIBEIRINHO et al., 2020).

The outdated way of detecting design errors is based on manually reviewing two-dimensional planning documents. This process is error-prone and time-consuming. BIM enables Automated Code Compliance Checking (ACCC) to evaluate a digital building model’s quality concerning its data, modeling, and design quality. All validation results can easily be documented and communicated between all stakeholders (Figure 1.1).

Checking for compliance with specific guidelines often requires the user to identify essential information to test for its claimed requirements and enrich building models with the determined additional data. Therefore, this master’s thesis outlines techniques to detect and structure this necessary information and approach an efficient mapping process. These issues are accomplished by utilizing recent BIM-tools for generating, processing, and evaluating mandatory data and checking results.



In Chapter 4 the basic knowledge of standards is explained and the relevant contents of MBO regarding this master's thesis are outlined. Additionally, norm's representation types and levels of complexity are postulated. Furthermore, different approaches of (semi-)automated information extraction methods are represented and evaluated.

The conceptual approaches of this master thesis are outlined in Chapter 5. In the beginning, the RASE mark up technique of the former chapter is applied and essential data for ACCC is outlined. The overall identification and mapping of additional data is generalized by terms of granularity and application of mapping tables.

Chapter 6 shows the implementation and validation of the before presented approaches. As case study, the paragraphs 33 and 34 of MBO are utilized as use cases. Existing and new developed Solibri Office checks are implemented to validate a test building model comprising several design errors regarding the underlying use cases. Furthermore, the approach is evaluated and limitations are discussed. Additional suggestions for future work of this thesis are reviewed in Chapter 7.

## Chapter 2

# Introduction to IFC, IDM, MVD and Co.

The basis of effective utilization of **BIM** are open data formats like **IFC** and determined methodology to structure new workflows, enabling operators to handle large amounts of information and data. This chapter covers basic knowledge of **IFC** and its general structure and outlines essential official methods for handling and enhancing **BIM** working procedures.

### 2.1 Historical Background

In late 1980, scientists began to search for consistent and loss-free data exchange, and the idea of a geometric and semantic representation arose. The first attempts at a loss-free exchange of geometric data between heterogeneous CAD-Systems were started in the 1970s, which resulted in Standard for the Exchange of Product Model Data (**STEP**). Due to protracted bureaucratic processes of International Organization for Standardization (**ISO**), these first steps of digitization in the **AEC** sector were not further pursued. By state-funded research and with the additional participation of industry, the standardization efforts were further intensified and, in 1995, substantiated with the founding of the non-profit organization International Alliance for Interoperability (**IAI**) which was later renamed buildingSMART International (**bSI**).

In 1997, **bSI** released **IFC** Version 1.0 as a standardized data format for exchanging semantic and geometric information, which is regularly updated and extended to the current version IFC4 (2021). One of **bSI** core strategies is its open access of **IFC** (Open BIM) and thereby guaranteeing a rapid rise of users and applications. Hence, it is available free of cost, vendor-neutral, and independent of ISO standardization (“Technical Roadmap 2021”, 2021, BORRMANN et al., 2015).

### 2.2 Industry Foundation Class

**IFC** is a vendor-neutral and standardized data format to describe and exchange information of buildings and civil infrastructures (“buildingSMARTa”, 2021). **IFC** is connected to the above mentioned **STEP** by having the same underlying data modeling language in common, named EXPRESS. EXPRESS is a declarative language and can be used to define object-oriented data models. The entity type is comparable to classes in object-oriented theory and can be additionally defined with several attributes and relations to other entity types. Furthermore, inheritance can be used to pass attributes and relations

to subtypes (BORRMANN et al., 2015). The following paragraphs cover the IFC Layers, their connected hierarchical structure, and object relationships.

## 2.2.1 IFC Layers

Due to the rising complexity and amount of classes, the IFC schema is subdivided into four layers. These layers structure the overall inheritance possibilities: upper layers can reference elements of lower layers, but not vice versa. Figure 2.1 highlights the four IFC Layers Domain Layer, Interoperability Layer, Core Layer, the Resource Layer, and its incorporated schemata which are explained in the following.

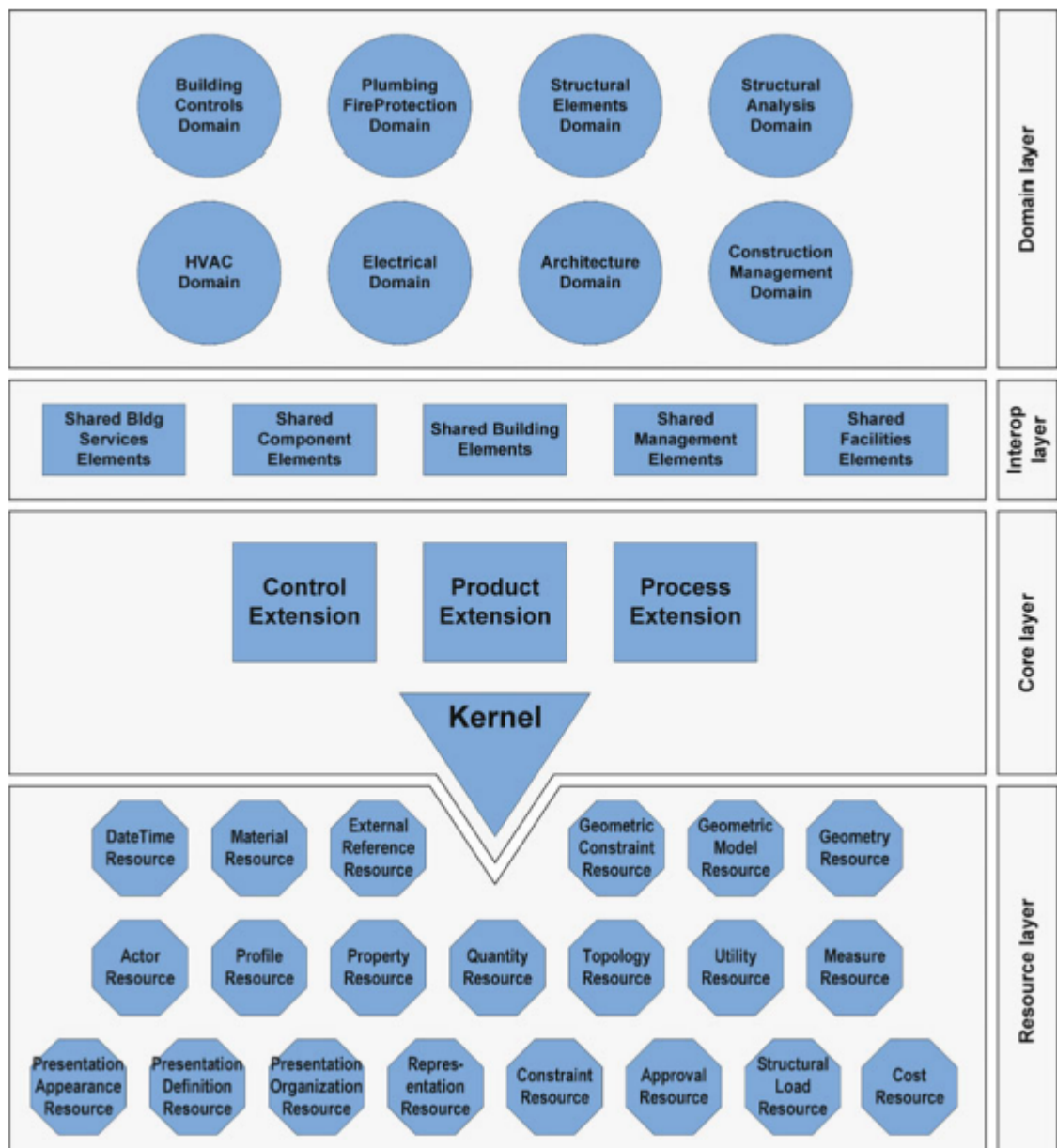


Figure 2.1: Four Layers of IFC Schema (“buildingSMARTb”, 2021)



### **Resource Layer:**

The Resource Layer is on the lowest level and contains the fundamental data structure, which can be referenced in the whole building model. The Resource Layer classes' uniqueness is the absence of identity, and it is impossible to instantiate a class as a discrete object. Only objects of the upper layers can reference these classes because the upper layer classes are sub-classes of *IfcRoot*. One of the most used schemata resources are, e.g., of type: Material, Topology, Cost, and Geometry.

### **Core Layer:**

The next level in the hierarchy is the Core Layer and its fundamental classes of the IFC schema. A central position is the Kernel-schema containing its abstract classes *IfcRoot*, *IfcObject*, *IfcActor*, *IfcProcess*, *IfcProduct*, *IfcProject* and *IfcRelationship*. Abstract classes can not be directly instantiated but by their sub-classes. *IfcRoot* is the start point of the inheritance schema of IFC. All these entities can be referenced and substantiated by the upper layers to define fundamental relations, basic structures, and general concepts. Additionally, the Core Layer contains three extension schemata *Product Extension*, *Process Extension* and *Control Extension* which are directly connected to the Kernel classes. These schemata offer classes to describe spaces and their relations, different processes, and the declaration of control objects.

### **Interoperability Layer:**

The Interoperability Layer, or Shared Layer, serves as an intermediate level and therefore contains classes derived from the Core Layer's Classes, which are often used in several domains. These definitions are used for inter-domain exchange of information ("buildingSMARTb", 2021). For example, *IfcWall* or *IfcColumn* are important components in the construction field but are used by several domains.

### **Domain Layer:**

The final level in the hierarchical layer structure builds the Domain Layer. Its domain-specific schemata cover classes that can be directly assigned to a particular domain and define entities of a specific product, process, or resource. These objects are used for intra-domain exchange and sharing of information. The current IFC4 implies 11 domains ("buildingSMARTb", 2021 BORRMANN et al., 2015).

## 2.2.2 Inheritance Hierarchy

The layer structure mentioned above of IFC builds the overall composition of the inheritance hierarchy in the object-oriented data format. This hierarchy defines which classes can access what kind of attributes and, therefore, strongly influence the specialization- and generalization relationships. Figure 2.2 shows some of the most important entities of the upper layers and thus, only picture a section of the whole underlying inheritance hierarchy. The red marked classes are abstract classes of the Kernel-Schema in Section 2.2.1 and are highlighted for clarity.

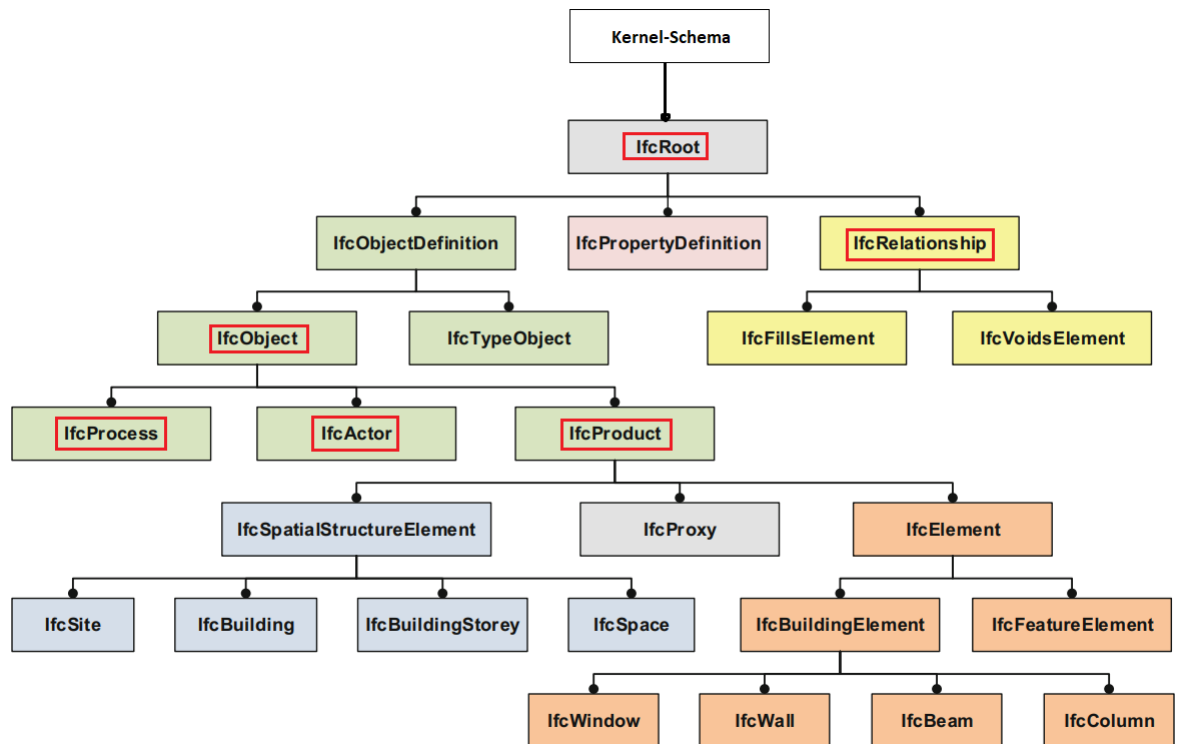


Figure 2.2: Most Important Entities of the Inheritance Hierarchy in the IFC Schema (BORRMANN et al., 2015)

The class *IfcRoot* directly or indirectly defines all entities, except for the ones belonging to the *Resource Layer*. All IFC classes are defined by bSI and additional information and explanation are available online. Listing 2.1 depicts the EXPRESS specification of the *IfcRoot* exemplarily:

Algorithm 2.1: IfcRoot EXPRESS Specification (“buildingSMARTc”, 2021)

---

```

ENTITY IfcRoot
ABSTRACT SUPERTYPE OF(ONEOF( IfcObjectDefinition , IfcPropertyDefinition ,
    IfcRelationship ));
    GlobalId :                IfcGloballyUniqueId ;
    OwnerHistory : OPTIONAL   IfcOwnerHistory ;
    Name :                OPTIONAL   IfcLabel ;
    Description :  OPTIONAL   IfcText ;
UNIQUE
    UR1 :                GloballId ;
END_ENTITY;

```

---

In Listing 2.1, the abstract supertypes *IfcPropertyDefinition*, *IfcRelationship*, *IfcObjectDefinition* are defined comparable to Figure 2.2. Additionally, all necessary and optional attributes are represented. Objects of *IfcRoot* require a unique identification by *IfcGloballyUniqueId*. Selective attributes of an object can be Name (*IfcLabel*), Information about ownership and origin (*IfcOwnerHistory*), and Description (*IfcText*) and are marked as OPTIONAL (BORRMANN et al., 2015, “buildingSMARTc”, 2021).

### 2.2.3 Object Relationship

In IFC, relations between objects are rather realized by the direct compound of these objects than by inserted objects which represent the relation by itself. The super-class of all relations is *IfcRelationship* and has 6 sub-classes. These sub-classes define different relations like spatial relations, e.g., the relations of spaces to surrounding components, defining multilayered components, implementing extension mechanisms like Property Sets, and different approaches of geometric representations.

#### Spatial Aggregation and Space-Element-Relation:

The spatial collection hierarchy in IFC is arranged by the super-class *IfcSpatialStructureElement*, and its sub-classes comprise semantics from the site’s representation to building, story, and spaces description. Additionally, in a lot of BIM applications, e.g., ACCC, the relation of spaces to surrounding objects, like walls, windows, etc., are of great interest. The sub-class *IfcRelSpaceBoundary* references via attribute *RelatingSpace* space objects and via *RelatedBuildingElement* particular elements. The resulting Space Boundaries always refer to the space object and can be distinguished in two levels. Level 1 neglects additional information, and Level 2 considers if on the other side is a space (Level 2b) or an element (Level 2a).

## Materials:

Another application of relationships is the connection of materials and components. An important functionality of IFC is the representation of multilayered components. The sub-class *IfcRelAssociatesMaterial* connects material to components, and composite materials can be linked to components by *IfcMaterialRelationship* as well. Additional material parameters can be coupled via *IfcMaterialProperties* which can use predefined property sets or specified by the user.

## Property Sets:

IFC covers a large variety of semantics and, therefore, must prevent reaching an unclear complexity. For this reason, the property definition of objects is split into static and dynamic (property sets) generated attributes. The former are fundamental features of objects, like for doors the *OverallWidth* and *OverallHeight*, and can be seen as standard information that always has to be delivered. Attributes, e.g., belonging to national standards, can be linked via property sets, using the sub-classes of *IfcProperty*. Additionally, property sets can be grouped via *IfcPropertySet* and referenced with objects via *IfcRelAssignsProperties*. A negative side-effect of this schema is a missing clear definition or labeling of certain objects and attributes. Different stakeholders may vary in labeling property sets which results in a lack of interoperability. Avoiding this problem, bSI introduced the buildingSMART Data Dictionary (bsDD) to publish standardized property sets definitions regarding classifications, properties, values, units, and even translations for different languages.

## Geometric Representation:

Because geometries are linked to an object, IFC handles semantic and geometric information separately. Therefore, objects can have several geometric representations, but semantic information is always leading: at first, an object is described via its semantic information and afterward by its geometry. The reason for this is the different application fields in the AEC sector with varying demands in terms of geometric quality. A widespread approach is the triangulated surface description which approximates the surface with a triangle mesh (*IfcTriangulatedFaceSet*). It is often used for pure visualization in applications because nearly every software can use the given information. But this comes with huge memory requirements, and curved surfaces are only poorly approximated. Solid Modeling (*IfcSolidModel*) is an approach often used by BIM tools to implement changes made by the user. A flexible method is the Boundary Representation or the Constructive Solid Geometry (CSG), which utilizes base units and boolean operations to construct geometries.

The implementation of geometric representations is always accompanied by positioning in space. The IFC schema uses local and global coordinate systems to enhance the flexibility in terms of changes. Objects inside a story are localized inside the coordinate system of the story, which is in turn placed inside the building's coordinate system. The global coordinate system refers to the *IfcSite* object, and all local coordinate systems are referenced (BORRMANN et al., 2015).

#### 2.2.4 Technological Outlook

IFC was implemented for a file-based exchange of building models. New technical concepts like Common Data Environment (CDE), Digital Twins, or future Smart Cities require an object-based use of IFC data and appropriate Application Programming Interface (API) designs to enhance interoperability. Therefore, IFC needs to be able to facilitate partial file exchanges and allowing smaller discreet exchanges. This new version of IFC would still enable the exchange of files and, additionally, the access, maintenance, and exchange via APIs. The current complex structure of IFC results from its closeness to EXPRESS and bounds it to the representation structure of STEP format. This complicates the process of accessing building models via an API and prevents the object-based access and exchange of partial IFC data.

bSI states in its technical roadmap of 2020 that IFC needs to ensure a reliable use by a stricter but also easier implementation to enable more predictable and consistent results. Due to rising requirements of additional domains and extensions, IFC has to become modular to provide efficient access. Additionally, circular references have to be removed to decrease its complexity, and the overall methodologies must base on modern computer interpretable languages like XSD, OWL, and JSON ("Technical Roadmap 2020", 2021).

### 2.3 Information Delivery Manual and Model View Definitions

The AEC sector is characterized by many different participating stakeholders in a project and, therefore, by a vast amount of exchanged data. bSI developed the IDM and the MVD to avoid issues in terms of data exchange by standardizing these processes. A fully enriched BIM-model contains a significant amount of information that is only required by specific stakeholders and thus is redundant in other use cases. Unneeded data might confuse users or interfere with a stakeholder's BIM use case and therefore lead to planning or construction errors.

The overall method is separated into two parts: the functional part, realized by the **IDM**, and the technical aspect, transposed by the **MVD**. This whole procedure is divided into six sub-processes:

1. Definition of participating stakeholders, their roles and tasks
2. Chronological and organizational integration of all sub-processes in Business Process Modeling Notation (**BPMN**)
3. Determining all data exchange interfaces
4. Formalizing Exchange Requirements (**ER**)
5. Information mapping onto **IFC**
6. Technical implementation of **MVD** via mvdXML

Sub-process 4 introduces the concept of exchange requirements which describes conditions of how to enrich certain elements in the **BIM** model. These boundaries are represented in tables, sorted by building elements, and include the chosen data type, units, relations, and sometimes permitted ranges of value. The tables have to be further formalized for an automated software implementation into the IFC data format (sub-process 5). Several **ER** can be grouped to an **MVD** to define mandatory or optional information and what kind of classes, attributes, and relations are used. **MVDs** enable partial mapping to allow the exchange of partial models for certain stakeholders with their specifically required information (BORRMANN et al., 2015).

**IFC 4** provides two basic views; the Reference View supports the coordination by easing the merging process of partial models or domain models. Only geometrically reduced objects combined with the relational representation of spatial and physical components are exported. The merging of models can be used for code compliance checking like error detection. In contrast, the Design Transfer View enables the export of advanced geometric components.

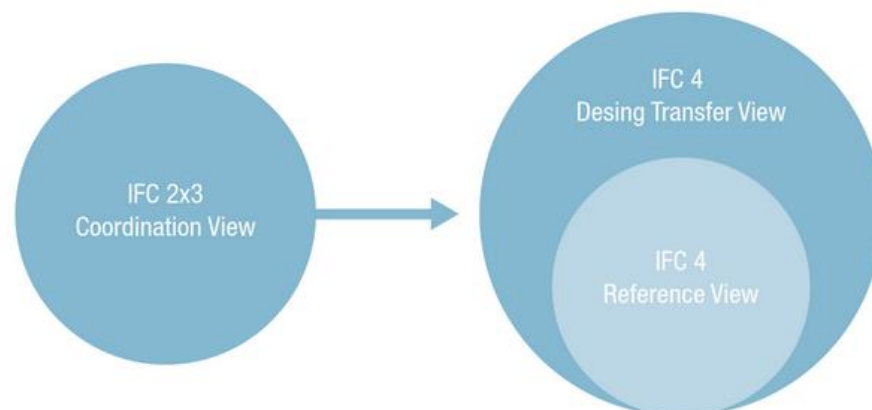


Figure 2.3: Scope of Design Transfer View and Reference View (IFC 4) Compared to Coordination View (IFC 2x3) (BALDWIN, 2017)

### 2.3.1 mvdXML

A huge amount of data requires possibilities to verify its correctness and completeness. For this reason, bSI developed the mvdXML as a technical implementation of the MVD. MvdXML is a standardized format to specify MVDs and its respective EIR, including exchange requirements and validation rules (CHIPMAN et al., 2016). These comprise exchange definitions of import and export scenarios via IFC. Exchange definitions represent a reduced building model according to the user's needs which eases the collaboration process between different stakeholders and assures an export of only relevant data. Additionally, it facilitates an automated validation of these building models regarding their data quality and replaces fault-prone manual efforts.

The mvdXML format utilizes Extensible Markup Language (XML) and can be created and edited in any text or XML editor. To ease this manual process, bSI provides the IfcDoc application for generating mvdXML files. The overall core structure of a mvdXML file begins with two main sub-elements: *Templates* and *Views*. *Templates* represent a list of reusable *ConceptTemplates*, which in turn define a graph, representing particular entities and their relating attribute requirements. These attribute requirements are defined inside a concept template and queried by *mvd:AttributeRules* and *mvd:EntityRules*. *Views* comprise a list of model view definitions (*ModelView*), which contain particular entities and relating concepts for defining exchange requirements. Every *ModelView* contains specific *mvd:ExchangeRequirements* and *Roots*. *Roots* can hold a set of concepts with template rules checking a subset of information (see Figure 2.4).

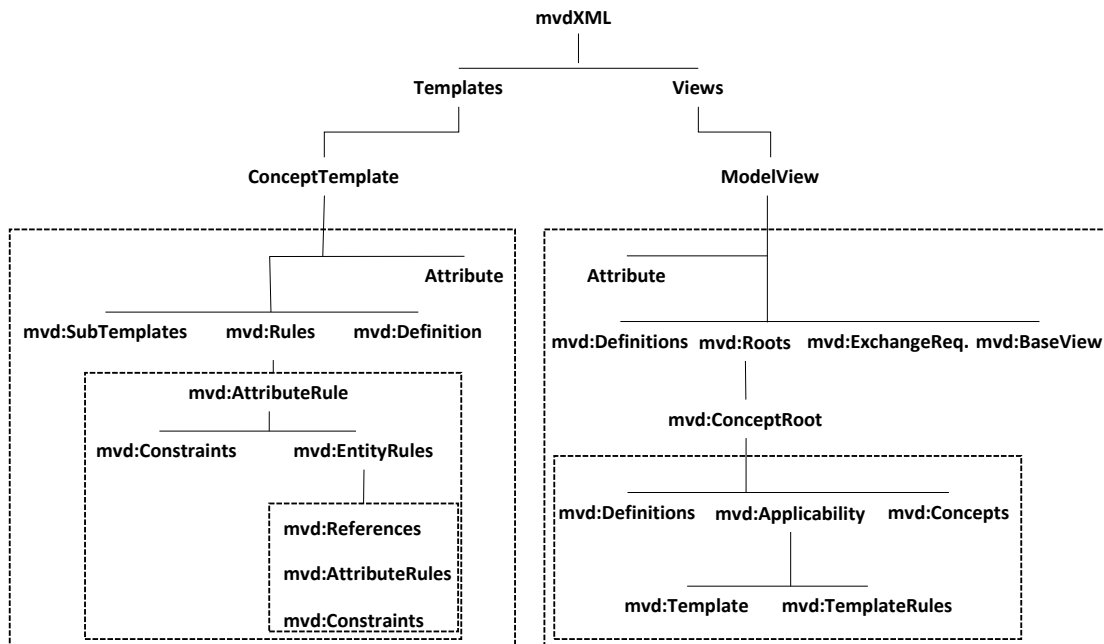


Figure 2.4: mvdXML Elements (Inspired by CHIPMAN et al., 2016)

An extraction of a mvdXML file is depicted in Algorithm 2.2 of a *Templates* body definition. The *ConceptTemplate* declares its attributes of an Unique Identifier (UID), name and the underlying IFC version. The applicable entity and root entity of this concept templates is

*IfcDistributionElement*. *mvd:Rules* includes the IFC definition of how a *IfcDistributionPort* is linked to the root entity *IfcDistributionElement*. The *IfcDistributionElement* is nested by several IFC entities and holds a reference to the relation class *IfcRelNests* which links it to *IfcDistributionPort*. In the *mvd:AttributeRules*, three attributes are declared and can be verified via the *RuleID* attribute (CHIPMAN et al., 2016).

Algorithm 2.2: Extraction of mvdXML File. Definition of *Templates* Body (CHIPMAN et al., 2016)

---

```

<Templates>
  <ConceptTemplate uuid="bafc93b7-d0e2-42d8-84cf-5da20ee1480a"
    name="Port Assignment" applicableSchema="IFC4"
    applicableEntity="IfcDistributionElement">
    <Definitions >
      <Definition >
        <Body>
          <![CDATA[<p>Distribution ports are defined by <i>
            IfcDistributionPort </i> and attached by the <i>IfcRelNests </i>
            > relationship. Ports can be distinguished by the <i>
            IfcDistributionPort </i> attributes <i>Name</i>, <i>
            PredefinedType </i>, and <i>FlowDirection </i>:</p>]]>
          </Body>
        </Definition >
      </Definitions >
    <Rules>
      <AttributeRule AttributeName="IsNestedBy">
        <EntityRules >
          <EntityRule EntityName="IfcRelNests">
            <AttributeRules >
              <AttributeRule AttributeName="RelatedObjects">
                <EntityRules >
                  <EntityRule EntityName="IfcDistributionPort">
                    <AttributeRules >
                      <AttributeRule AttributeName="Name" RuleID="Name"/>
                      <AttributeRule AttributeName="PredefinedType" RuleID
                        ="Type"/>
                      <AttributeRule AttributeName="FlowDirection" RuleID
                        ="Flow"/>
                    </AttributeRules >
                  </EntityRule >
                </EntityRules >
              </AttributeRule >
            </AttributeRules >
          </EntityRule >
        </EntityRules >
      </AttributeRule >
    </Rules>
  </ConceptTemplate>
</Templates>

```

---



MvdXML format represents an automated way of defining and assuring a model's data correctness and existence. bSI implemented the format to facilitate reusable templates and thus, implementing an efficient way of data quality checking. MvdXML lacks in standardization due to its high flexibility of creating *ConceptTemplates*. This high level of flexibility increases the chance of multiple identical mvdXML files for a similar or identical use case and decreases its efficiency and, therefore, its applicability. Additionally, it requires knowledge of the XML format and can result in an unclear structure that is hard to control, especially in terms of manual adjustments due to the deficient quality provided by IfcDoc (POPGAVRILOVA, 2020).

## 2.4 LOD, LOG, LOI

A project's BIM model relies on its data which all stakeholders integrate. The amount of data steadily increases during the progressing project phases and results in final status, representing all necessary building characteristics. To ensure the data quality of building elements in all project phases, the LOD was implemented. LOD establishes different levels of information and covers requirements for geometric (Level of Geometry (LoG)) and semantic (Level of Information (LoI)) information to guarantee a distinct and reliable data basis. The particular LODs are determined in the BIM Execution Plan (BEP) and progress with relating work phases, e.g., Fee Structure for Architects and Engineers (HOAI). The "BIM4INFRA", 2020 imposes five levels of development and its underlying working stages (see illustration Figure 2.5):

LOD 100: Environment and rough building dimensions

LOD 200: Generic objects with approximated quantities, size, shape and location

LOD 300: Specific objects with reliable quantities, size, shape and location

LOD 350: LOD 300 + object relations

LOD 400: LOD 350 + increased level of detail (e.g. for fabrication)

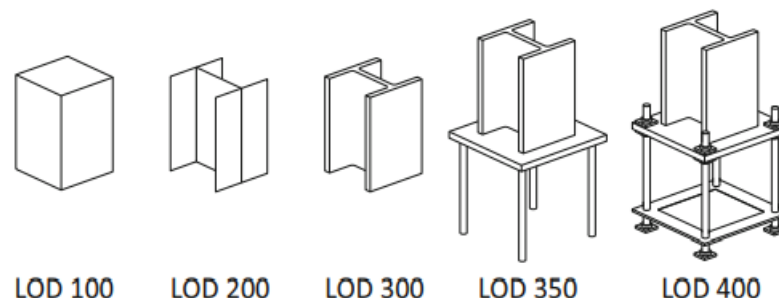


Figure 2.5: Level of Development Illustrated by the Example of a Steel Beam (BORRMANN et al., 2015)

The detailed requirements concerning geometry and semantics have to be defined for every underlying use case. The intention behind LODs indicates the need for data management due to its huge amounts. Another important component is a coherent nomenclature of information that all stakeholders agreed on and used in their data enrichment processes. bSI aims to ease this issue by creating bsDD, an online service that provides a database to standardize additional data for particular use cases. Currently, bsDD can be accessed via an API but bSI plans to improve its usability and provide new functionalities like integrating bsDD content into Information Delivery Specification (IDS) contracts (“buildingSMARTd”, 2021, “BIM4INFRA”, 2020).

## 2.5 Summary

IFC and BIM exist over 20 years and reformed the way of planning in the AEC sector. IFC is structured by a four-layer system which facilitates a strong inheritance hierarchy. This comprises the definition of building elements, assignment of additional attributes and properties, like materials, and its relations to other building elements. The current approach of IFC reaches its limits and will be reworked over the next coming years to facilitate upcoming technical concepts like Digital Twins or CDE. Due to its high amount of information and different stakeholder, a good BIM workflow requires management processes that addresses the client’s interests and assure high quality at the same time. For this reason, IDM and MVD are one of several management and technical implementation concepts to solve these issues. These methods address the procedure of data exchange and are implemented via the mvdXML format. In addition, the principle of LOD helps to determine the required depth of information in a particular project phase. Therefore, a semantic and geometric category is distinguished and represented by five different LoG and LoI levels. A general understanding of central characteristics and concepts of BIM and IFC is required to structure and implement efficient code compliance checking procedures.

## Chapter 3

# Code Compliance Checking

Chapter 3 gives a brief introduction of the process of code compliance checking and its place in a **BIM** workflow. For this purpose, the different quality levels of building models are defined for a deeper understanding of the overall compliance checking task. In a final step, different technical implementations of code compliance checking approaches are outlined and reviewed regarding their advantages and limitations.

### 3.1 Introduction

The implementation of **BIM** in the **AEC** sector shifts the planning processes to a model-based workflow. The building model is the central object in all planning cycles and contains the required information. A main use case of **BIM** is the direct generation of plans out of the underlying building model. Therefore, checking the model's quality and ensuring a consistent model is mandatory for a proper workflow. Automating these manual and iterative compliance processes results in improved efficiency and profitability.

Code compliance checking processes generally describe methods to test for a model's quality and are structured as shown in Figure 3.1. The first procedure is the rule interpretation, which can only be realized by translating regulatory contents into a machine-interpretable language. Due to various representation styles of standards or legal texts, standardization of this process is still a challenge. This step results in a verifiable rule which is executed subsequently. This execution comprises the interpretation and processing of information delivered by the building model. The preparation of a building model is mentioned in Figure 3.1 because of possible inconsistencies or missing information in the underlying building model. The prevention of these errors can be realized via the implementation of preprocessing steps, like modeling guidelines or detection by data compliance checks. Testing for consistent models is a major task in utilization of **BIM** and has to be applied in various use cases. The final step of a code compliance checking process is to prepare and preserve checking results for a user review. Therefore, code compliance checking covers the detection of planning or modeling errors and provides possibilities for better communication between stakeholders but does not comprise its elimination (PREIDEL, 2020, PREIDEL and BORRMANN, 2015).

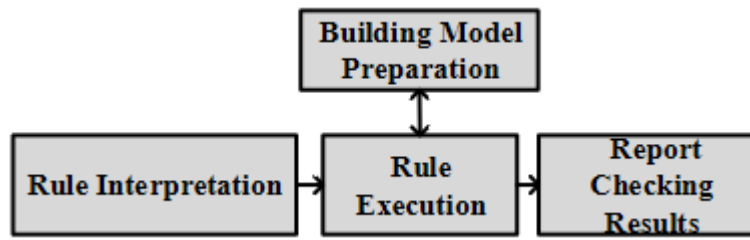


Figure 3.1: General Structure of Compliance Checking (PREIDEL and BORRMANN, 2015)

## 3.2 Levels of Model Quality

Section 3.1 outlined the general need and structure of code compliance checking processes. Checking a model's quality can be distinguished in several levels, which vary in its person in charge and overall testing scenarios. Generally, the issue of considering a model's quality can be divided into three levels.

The initial point of every building model is its design in a modeling tool and its resulting data quality. The provided model must be checked in terms of overall correctness and syntax. In addition, the raw data must be delivered in the determined project milestones, which can also be checked in fixed data drop scenarios. The level of data quality depends on software interfaces, such as the export into data formats like IFC. The software vendor must ensure the correct and complete export of data into an open data format. This issue is intensified by bSI and its certification of BIM software. Additionally, the model's author is responsible for his delivered data by modifying export settings which can enormously influence the overall quality for particular use cases. An example of the abundance of data quality is the securing of data availability during specific project milestones. This provided data must be complete and represent the current planning state to guarantee flawless data processing.

If the first level of model quality is guaranteed, content-related quality has to be taken into consideration. This level comprises semantic and geometry-related data and its interaction. Semantic and geometric information depends on various arranged requirements concerning the underlying project and model use case. Compliance with these issues must be ensured by the responsible model author or BIM-coordinator.

Based on the before mentioned levels and its provided data, use cases concerning the planning quality of engineers and architects can be determined and developed. The verified contents result from norms and regulations, but also from client requirements which were agreed upon in the BEP. All stakeholders are responsible for their introduced data and, therefore, are in charge of its compliance. Important is the difference between data or content-related quality and planning quality. These two fields can independently be flawless or incorrect of each other, but the former builds the required basis for the latter. The establishment of a general way of compliance checking to ensure a high-quality model is not possible and always depends on the underlying use case (PREIDEL, 2020).

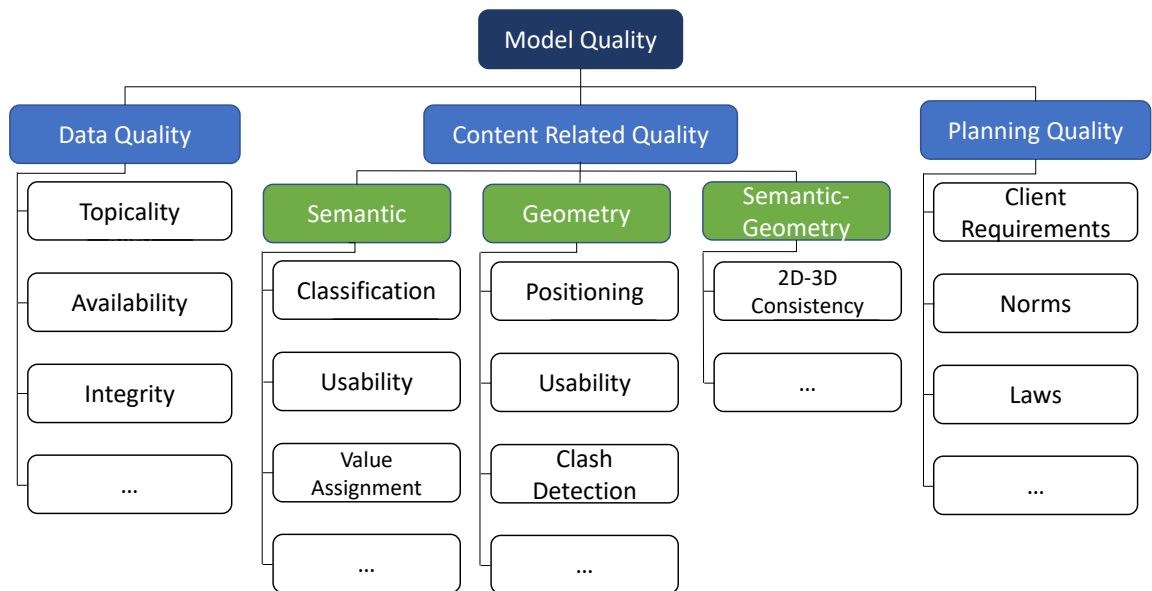


Figure 3.2: Model's Quality Levels (PREIDEL, 2020)

### 3.3 Technical Approaches

The before outlined process of code compliance checking is a mandatory method for a BIM workflow and can support its user in terms of efficiency and accuracy. Several different approaches of technical implementation were developed to counteract particular weaknesses of already established approaches in the past. In the following, selected code compliance checking approaches are depicted and reviewed in terms of their strengths and weaknesses.

#### 3.3.1 Hard-Coded Tests

Hard-coded tests represent programmed tests that are later immutable and check a particular use case for its compliance. Software vendors often provide these tests but as well, can be developed via an open API by software developers and engineers. This approach is one of the most widespread technical approaches and enables a fast and cheap compliance procedure. But, on the other hand, these tests represent black-boxes for the user, and detailed insight into its functionalities is not assured.

Figure 3.3 depicts the difference of a black-box solution compared to a white-box approach. Referred to code compliance checking, the underlying building model represents the input data, and the results of the compliance test stand for the output. In between, the overall checking process of applied test scenarios is illustrated by the box. In a black-box approach or hard-coded compliance test, the overall functionality and checking strategy remains invisible for the user. In contrast, all proceeding processes are observable in a white-box approach.

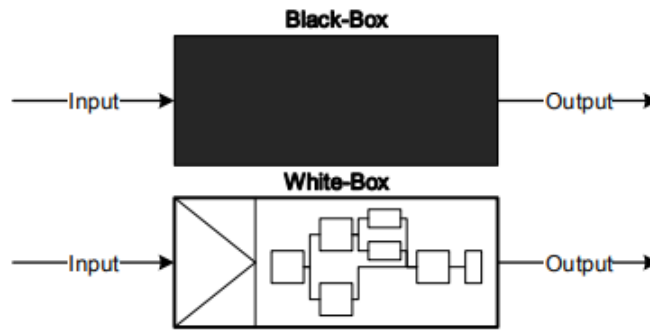


Figure 3.3: Schematic Diagram of Black-Box and White-Box Processes (PREIDEL, 2020)

This matter is a significant weakness and risk of the hard-coded compliance test. Incorrect use or misinterpretation of a test's functionality can result in undetected errors and, thus, planning and construction errors or delays. Furthermore, creating compact and highly automated code compliance tests can result in lacking user involvement. However, the user remains the person in charge and must supervise all checking results due to their accuracy. Additionally, the development of coded compliance tests requires highly skilled engineers, which are rare in the current market situation. This low supply of capable labor forces limits the integration of a digitized compliance checking process and forces up prices.

In order to counteract the before mentioned problems and challenges, libraries of small complementing tests reduce the misunderstanding of tests and improve the user involvement during the compliance checking procedure. Every test contains simple scenarios, and the user builds the overall test structure by himself. All applied tests are described in detail in a test documentation concerning their functionalities, required data, and limitations. These strategies can ensure an efficient and accurate compliance process using hard-coded compliance tests in an already widespread technique. To further accelerate the implementation of code compliance checking, a new field of engineering could be established in the [AEC](#) sector. Supplying the market with high quality and standardized compliance checking libraries by private or governmental services would have the potential to fuel the implementation process of [BIM](#). Many planners require periodic and highly generalizable use cases, and this offer would decrease the pressure on offices and companies for rare, well-educated engineers with coding knowledge.

The following will outline two different approaches of hard-coded compliance test software. *Solibri Office* represents an example of a code compliance checking tool established in the private market and used by engineers and architects. The latter is a tool developed by building authorities to enable rapid implementation of flawless [BIM](#) methodology matching high-quality standards.

### **Solibri Office**

*Solibri Office* was released in 2000 by the Finnish software company Solibri Inc., which is now part of the Nemetschek Group, to enable reviewing and checking digital building models for their quality. It is a popular and important tool for code compliance checking in the AEC sector and can be used as a BIM model viewer, but also provides functionalities like classifications. These classifications cover standard classifications like *OmniClass*, and additionally, enables individual classifications as well. All results can be visually reviewed and can be utilized in an efficient code compliance checking process. In terms of code compliance checking, basic rules are delivered by Solibri and can be adjusted via the Ruleset Manager. These rules enable engineers and architects to check building models for the existence of particular attributes and their relating data types (data quality), or overall planning quality like escape routes.

Adjusting the underlying tests concerning particular parameters assures enhanced user involvement. In addition, gathering specific tests as a set of rules enables the assignment of conditional operations between the tests and, therefore, guarantees the usage of overall small single tests, decreasing the chance of misinterpretation or use of large fully automated compliance tests. Currently, an API was released to enable the user to develop his compliance tests to extend the existing rule library by individual requirements. Regardless of its user-friendly functionalities, the use of the provided tests and the development of compliance tests require high coding skills and a large amount of knowledge concerning the data structure of IFC and Solibri. Additionally, the black-box character can not be eliminated and, therefore, remains a risk of undetected errors (PREIDEL, 2020).

### **CORENET - Construction and Real Estate Network**

In 1995, the Building Construction Authority (BCA) of Singapore established the Construction and Real Estate Network (CORENET) and thus, was one of the first national digital platforms digitizing the planning process in the AEC sector. CORENET was introduced to enable the submission of documents digitally and speeding up the overall authorization process. A tool of CORENET enabled a first automated compliance check for fire safety and accessibility of 2D plans. This approach of ACCC was continuously developed, and the functionality of using 3D building models was introduced by the implementation of the open data format IFC in 1998. The foundation of the current *e-plan Check* software was laid in 2002, which uses hard-coded tests and a developed external library, called FORNAX. This provides an API for an autonomous test development (EASTMAN et al., 2009).

CORENET represents the possibility for state authorities to fuel the overall digitization process successfully. In 2007, the authorization process took about 103 days and was reduced to 26 days in 2014 (FIEDLER, 2015). Additionally, many projects are awarded by public authorities, and planning errors can result in high additional costs funded by tax revenues. The participation of state authorities in a project's compliance process can economize tax revenues and reduce the waste of tax money.

The strong black-box character of **CORENET** results from the underlying hard-coded compliance tests, which are not accessible for the user. Therefore, the overall functionality is not observable and is characterized by low user involvement. The compatibility with *FORNAX* strongly relies on the quality of the underlying data, which can result in complications (SOLIHIN et al., 2016). In addition, the extension via the **API** requires extensive coding knowledge and hinders it in a widespread utilization (PREIDEL, 2020). Regardless of its downsides, **CORENET** is an effective approach of authorities to enhance its administration efficiencies and indicates the strengths of digitization tools.

### 3.3.2 Domain Specific Programming Languages

The issue of user involvement regarding compliance procedures via hard-coded tests can be reduced but not eliminated. Enabling the user to develop and implement the overall test scenario facilitates a high user engagement. Domain-specific programming languages come into place to advance this problem. It represents a programming language utilized by the user for overall test development. A unique feature is the preparation and incorporation of data that the underlying regulation or data model does not comprise. Preparing and incorporating this data is facilitated by more considerable freedom for formulations, which amplifies the risk of inconsistencies in the language's declaration at the same time. The syntax has to be determined in a particular scale which is represented by the underlying domain. In addition, a domain-specific programming language should be expressible by a simple syntax to assure the possibility of widespread use (PREIDEL, 2020). The following outlined approach is an example of a domain-specific programming language and was designed and implemented by Jin Kook Lee to evaluate building circulation and spatial programs.

#### **Building Environment Rule and Analysis**

LEE, 2011 developed Building Environment Rule and Analysis (**BERA**) for the translation of complex regulation contents. **BERA** is an imperative language and was designed to provide ease of use for the complex underlying process. Therefore, a simple syntax is implemented to quickly and easily access information via a dot notation. In addition, the **BERA** Object Model (**BOM**) is introduced and based on the **IFC** data scheme. The use of **BERA** is restrained to spatial and geometric information, and so is the underlying **BOM**. Similar to other programming languages, **BERA** can be extended by other functionalities and objects and, therefore, extend its use to other compliance fields.

The syntax was designed according to 4 core elements: *Reference Directives* enables an import of external libraries to extend functionalities. Principle of *Object Model Definition and Declaration* facilitates the declaration of objects and enables access of these classes, which can be accessed by dot notation and its comprised information. In addition, logical operations and advanced functionalities like recursion, inheritance, and more are applicable. To facilitate the major use case of testing for complex regulation contents, **BERA** provides a *Rule Definition* functionality. This is provided by a Rule object and an *Execution*



*Statement* grants access to the before defined information (see Algorithm 3.1 and 3.2). The connection to Solibri Office realized a graphical representation of the results (LEE, 2011).

Algorithm 3.1: Definiton of myrule in BERA (LEE, 2011)

---

```
Rule myrule(Space space) {  
    space2.area > 1000;  
    space2.Floor = "Level_1";  
    space2.security = "public";}
```

---

Algorithm 3.2: Definition and Query of Space Category midOffice in BERA (LEE, 2011)

---

```
Space midOffice {  
    Space.area > 600;  
    Space.area < 900;  
    Space.height > 9;  
    Space.name = "office";  
    Space.name != "shared";}  
get(midOffice);
```

---

**BERA** depicts the potential of domain-specific programming languages concerning code compliance checking. Nevertheless, imperative languages require a high level of generality in their logic base to define complex compliance processes. This missing versatility restricts a wide use in the general field of code compliance checking (PREIDEL, 2020).

### 3.3.3 Visual Programming Language

Visual Programming Language (**VPL**) are formal languages with a graphical notation that use graphical elements for describing operations and functions instead of textual entities. This approach forms a system consisting of symbols and application rules. Every symbol represents a visual element and conveys an explicit meaning that enables intuitive distinction with little knowledge. A distinct graphical difference between these elements facilitates a clear visualization and interpretation by the user. Additionally, its flow-based character amplifies the high user-friendliness. The developed system of visual elements forms a type of information flow, starting with the initial information as input to its further processed information inside the graph to the final user-defined output data. These characteristics make it a user and beginner-friendly language tool. One of the most widespread **VPL** approaches is *Grasshopper* which is a Plug-In for the 3D modeling software *Rhinoceros 3D*. *Grasshopper* extends functionalities of *Rhinoceros 3D* by specifying different functionalities via a visual graph (see Figure 3.4).

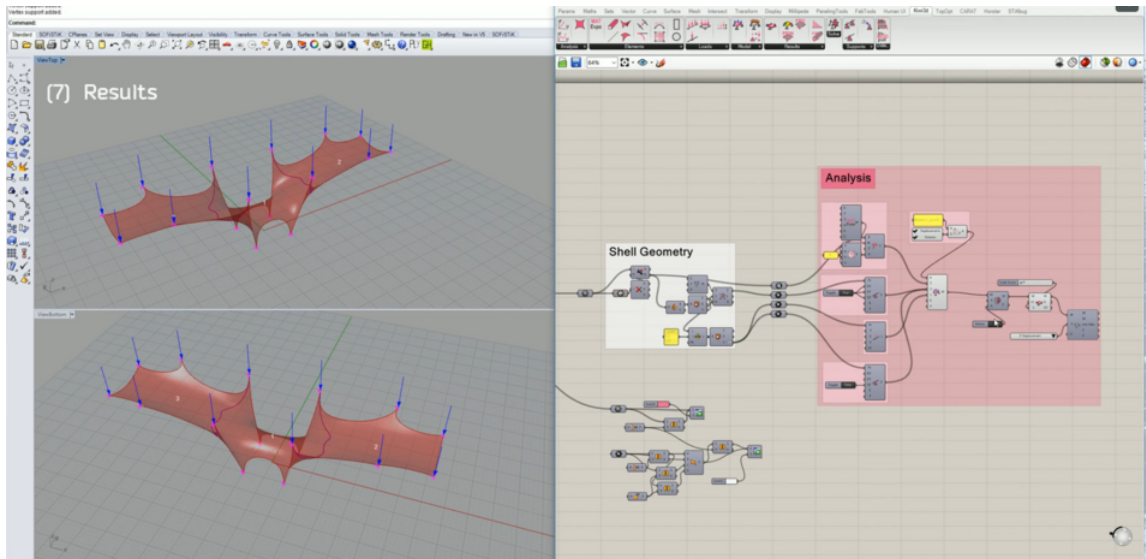


Figure 3.4: Form Finding and Static Analysis of a Bridge via Kiwi!3d (Plug-In for Grasshopper) “TUM\_ST”, 2021

High user interaction and ease of operation are one of VPL’s core advantages and are the reason for its widespread usage. Though, complex tasks can result in unclear and incomprehensible graphs and, therefore, can decrease its usability. In addition, functionalities like loops and recursion are core principles in many programming languages and enable the implementation of complex issues. Many approaches of VPL do not support these operations or are only hardly implementable (PREIDEL, 2020). The following outlines an approach developed by PREIDEL, 2020 to enable the use of VPL in a code compliance checking process. Its core structure, and strengths and weaknesses are reviewed to evaluate its contribution to a user central compliance checking strategy.

### Visual Code Checking Language

PREIDEL, 2020 developed VCCL to address the above-outlined approaches’ weaknesses by enabling a high user involvement in a code compliance procedure. To assure VCCL is an applicable and user-friendly programming language, three core principles were identified to facilitate this concept. *Domain Specific Applicability* describes the requirement of low barriers relating to its practicability by domain-related labor forces. A low prerequisite of coding knowledge reduces these barriers. Every coding language is implemented in a particular application field and must meet its user’s needs. A *Domain Specific Expressiveness* concerns the language design and thus its graphical elements and its resulting grammar to match the underlying use case and its domain-specific requirements. In addition, the principle of *Transparency and Verifiability* allows a high user-machine connection and hedges this error-prone process.

VCCL is a strongly standardized and object-oriented language. Its graphical elements represent particular objects and methods and their provided ports. A method represents a data processing instance for one specific task, and ports serve as data transmission of certain data types. PREIDEL, 2020 determines VCCL’s properties by being highly generic to enable a broad application of its elements on various use cases independently

of its complexity. This assures the general applicability of its graphical methods to any compliance checking scenario. This general applicability can only be achieved by a great extent of granularity. Transparent code compliance checking processes rely on a method's decomposability to enable the user to identify all sub-processes and thus feasible arising problems. The user plays a vital role in the application of VCCL and acts as an additional controlling instance. Therefore, VCCL only aims to achieve a semi-automated compliance checking process. The underlying data model strongly influences the language's complexity. Due to its widespread utilization in the export and exchange of building models, IFC is utilized as a basis. The hierarchical structure of building elements, storing attributes, and use of objective relations is adopted. Complex but often applied principles of IFC, like inverse attributes, were discarded and displaced by the use of references to its particular entity.

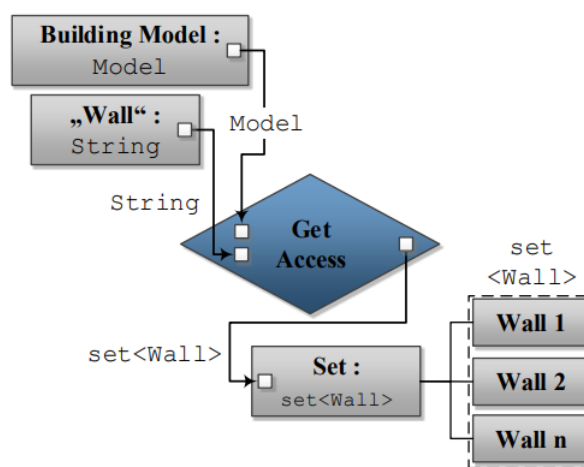


Figure 3.5: VCCL Graph Describing the Access to a Set of Walls. The Blue Trapezoid Represents the *Get Access* Method and Grey Rectangles Underlying Objects. (PREIDEL and BORRMANN, 2015)

The underlying functionalities of methods, like the *Get Access* function of Figure 3.5, are realized by atomic methods. These atomic methods build the basis of the VCCL library and are inaccessible for the user. An assembly of several objects and method elements connected via provided ports results in a user-defined compliance checking process. Assigned ports enable the input and output of data, and VCCL permits the production of various output information. This functionality reduces the overall amount of knots and edges and, therefore, reduces the overall complexity of the graph. Facilitating complex tasks like code compliance checking, the VCCL library comprises numerous different fields of operations. For example, comparing and applying arithmetic operations, or filtering for building elements and processing its relation, are core principles for a proper compliance procedure. Additionally, it provides methods to evaluate preliminary results and control structures like iterations, solved by loops and bifurcations via *if-else* structures. These functionalities assure a good user-machine interaction and enable the implementation of complex compliance programs. Figure 3.6 depicts a graph for compliance testing of the DIN 18232-2:2007-11.

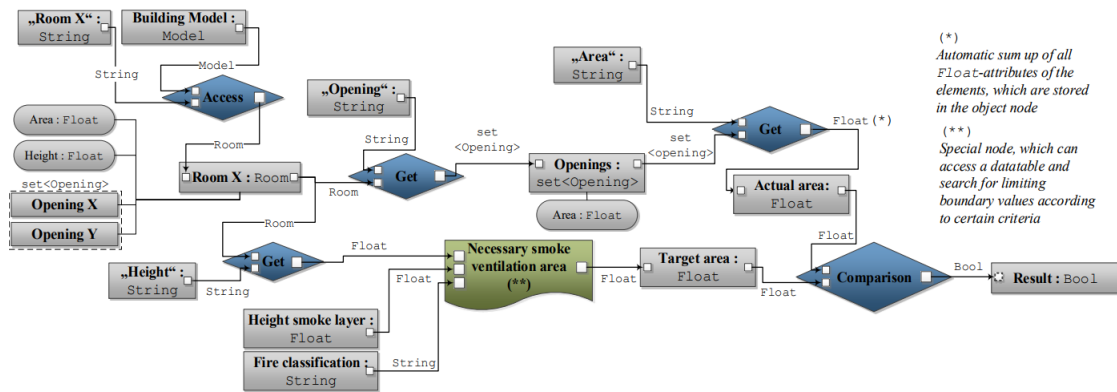


Figure 3.6: VCCL Graph Describing the Central Regulation of DIN 18232-2:2007-11 (PREIDEL and BORRMANN, 2015)

VCCL is a promising approach of a user-central code compliance checking process. Nevertheless, its expressive power depends on the total availability of atomic methods. For this reason, the outlined system requires an extension of the existing library to widen its field of application. However, this expansion of functionality comes with many methods, resulting in a decrease in clarity and transparency. In addition, it decreases user-friendliness, especially for beginners, and therefore, could prevent widespread usage. Another limitation is the high labor costs for simple compliance checking queries (PREIDEL, 2020).

### 3.4 Summary

In conclusion, it is vital to understand the importance of code compliance checking in a BIM based project. Understanding the general structure of a compliance checking process is the basis of comprehending the importance of a proper rule interpretation procedure and the decent preparation of the underlying building model. A building model's quality depends on three levels, of which the basic level represents the delivered data quality. The intermediate level comprises the semantic and geometric related data and its interaction. The third level represents the planning quality incorporated into the building model. These different quality levels are subject of the code compliance checking process and can be validated by different technical approaches. The before-reviewed hard-coded tests represent the most widespread approach but lack in their level of user involvement and transparency. These issues can result in undetected errors and, therefore, in high additional costs and time delays. Two software approaches are outlined and represent applications of the private business sector (*Solibri Office*) and public authorities (*CORENET*). Both software embody a high black-box character and thus increases the risk of undetected model or planning errors. The other two technical approaches address the limitation of low user interaction by implementing a domain-specific programming language and a visual programming language. The former, *BERA* relies on a typed programming language, using simple dot notation to facilitate the development of code compliance checks. On the other hand, *VCCL* uses graphical elements for a code compliance check implementation.

Both approaches facilitate a user-central compliance checking process but require a large library extension for more widespread usability. This need for an extension increases its complexity and complicates its access for beginners.

Incremental test scenarios utilizing hard-coded compliance tests can decrease the black-box character and enable efficient and fast compliance checking procedures. Therefore, this work implements smaller single tests via the Solibri Office [API](#) in combination with detailed test documentation to extend already existing compliance tests to validate digital building models.

## Chapter 4

# Musterbauordnung

Norms and regulations are widely spread in various economic sectors to guarantee uniformity and thus assure productivity while complying with quality standards. This chapter outlines the contents of the fifth section of [MBO](#) and categorizes it in terms of complexity and type of representation. In addition, the last section depicts a selection of information extraction methods of norms and regulations.

### 4.1 Introduction

Standards are common guidelines in all economical sectors and are applied to guarantee certain rights and obligations. The two most significant stakeholders in distributing and manifesting laws or guidelines are various industry associations and the State itself. Concerning Germany, three different types are distinguished: German Building Regulations, norms, and technical or rather customer-specific guidelines. The former covers private and public building laws and are passed by the German State. These laws have legal character, and all residents in Germany can refer to them. Additionally, *Deutsches Institut für Normung e.V.* publish norms to ensure products' and processes' quality, irrespective of where or by whom it is produced. Guidelines comprise special field specifications and are published by different associations or national facilities. Norms and guidelines are only compulsory in terms of application if specific laws refer to them. Due to representing the current stage of technology, norms and guidelines can be applied in court proceedings regarding the warranty of defects (BLIND et al., 2021, "VDI", 2021, PREIDEL, 2020).

### 4.2 Fifth Section of Musterbauordnung

Germany follows federalism's organizational principle, and every federal state has its own state-building regulations (Landesbauordnung). To ensure standardization, the conference of the Minister of Construction publishes and updates the [MBO](#) on which all state building regulations are based. The [MBO](#) itself does not have a legal status, which only comes into force by implementation in state-building regulations ("DIBt", 2021).

The MBO comprises requirements concerning:

- Developed properties,
- Engineered structures,
- Project's stakeholder,
- Building supervisory authorities and its procedures.

This thesis covers *Chapter 5: Emergency Routes, Openings, and Protection Devices* (§§ 33 - 38), which regulates fundamental fire protection standards of building elements. In § 14, MBO defines its fire protection goals by rescuing humans and animals, enabling firefighting operations, and preventing the spreading of fire and smoke. For these protection goals, §33 comprises requirements for the escape ways of a building, which are also used for firefighting operations. Spaces are supposed to have two escape ways directing to open air. The first escape way leads over a necessary staircase, and the second escape way can also use fire brigade rescue equipment. The latter is only required if no safety stairwell is present. The following paragraphs § 34 - 38 include further demands on stairs (§34), stairwells (§35), corridors (§36), windows, doors, and openings (§37), and barriers (§38). All these elements can be part of an escape route. They must meet strict requirements concerning fire behavior of load-bearing elements and coverings and sufficient sizing of components that serve as a part of the escape route. In consideration of special structures like schools, hospitals, high-rises, etc., further requirements are mandated in several special building regulations (PLUM, 2016, "Musterbauordnung", 2019). In the following, the MBO's contents of the fifth section is outlined more in detail:

### **§ 33 - First and Second Escape Way**

Paragraph 33 establishes the requirement of two independent escape ways leading to open air. The MBO differentiate horizontal and vertical escape ways. Both horizontal escape ways can use the same corridor, but a necessary stair must be used as the first vertical escape way. The second vertical escape way is required for buildings with common rooms and can either be a window if the fire brigade has the required equipment or another independent stair. A second vertical escape way is always required except for the presence of a fireproof stairwell (security stairwell) (MAYR, 2014).

### **§ 34 - Stairway**

This paragraph defines necessary stairways as stairways connecting two stories and are part of an escape route. The MBO excludes particular types like escalators or retractable stairways. Additionally, claims regarding the flammability properties of the stair's bearing structure are defined, and the existence of railings is postulated. An adequate usable width of stair flights or distance from a door opening in the stairways direction is arrogated but not further specified or limited by further details.

### **§ 35 - Necessary Stairwell and Exits**

All necessary stairways (§34) must have a necessary stairwell except for building classes 1 and 2, specific maisonette apartments, or external stairways. Paragraph 35 also limits the escape routes' length to 35m reaching a necessary stairwell from every location in a common room. Concerning super-imposed basements, two exits to necessary stairwells per story are required, and for multiple stairwells, an adequate distribution should be ensured. The request for a direct way to open-air is also taken up, and if not fulfilled, the following room with determined properties must be built. As well as for stairways, the flammability properties of materials and building elements is specified in detail as well as for all openings. Additionally, all necessary stairwells must have lightning.

### **§ 36 - Necessary Corridors and open Corridors**

The MBO introduces the concept of necessary corridors as corridors that are part of an escape route and claims particular cases that do not require necessary corridors. All regulations of paragraph 36 only concern necessary corridors in terms of their human traffic capacity, length of smoke sections, openings, and flammability of building elements and materials.

### **§ 37 - Windows, Doors and other Openings**

Paragraph 37 outlines minimum dimension requirements for windows and doors and claims the existence of at least one opening leading to open air in basements to ensure smoke removal.

### **§ 38 - Barriers**

In the last paragraph of section 5, the need for barriers is explicitly outlined, and minimum barrier dimension requirements regarding certain building heights are claimed.

## **4.3 Representation & Complexity of Regulatory Requirements**

### **4.3.1 Representation**

A norm's content can be described by a prescriptive or performance-based approach and differ in specifying the implied requirements. The former assesses the included contents and defines, e.g., if a specific material is permitted. It claims the desired performance and aim, but does not state a concrete approach. On the other hand, the performance-based way just generally states, e.g., in case of fire, the use of a necessary stair must be possible for a sufficiently long time, without referring to permitted materials. Therefore, in the planning phase, it must be demonstrated that all requirements are met. This elaborate proof demands additional effort from all stakeholders and is much more complex to translate into computer-readable information than a prescriptive represented norm. A fundamental problem of the prescriptive approach is the restraint of new and enhanced products due to a missing admission. New technical solutions need special application and



permission (PREIDEL, 2020). In regard to the MBO, it has a strong performance-based character because it serves as a bill to be transferred to the law of federated states.

Additionally, the MBO is a continuous text, thus does not include other types of representation like diagrams, graphics, tables, or equations. Continuous texts follow certain principles and rules concerning the structure and written declaration defined in DIN 820-2 ("DIN 820-2", 2021). A principle is the usage of modular auxiliary verbs to precisely describe the content of standards and minimize misapprehension. Auxiliary verbs apply to requirements (must), permissibility (is allowed), possibility (can), and recommendation (should) and its related negations (MATTIUZZO and MIESNER, 2021). The conference of the Minister of Construction aims to set strict guidelines; thus, MBO contains no recommendations. Nonetheless, MBO does not precisely specify all demanded requirements and leaves scope for interpretations:

§34 Treppen

*5) Die nutzbare Breite der Treppenläufe und Treppenabsätze notwendiger Treppen muss für den größten zu erwartenden Verkehr ausreichen.*

The quote mentioned above is only one example of several vague statements in the fifth section of MBO. The performance-based requirement states that a stair's width must match the maximum expected traffic without claiming particular values. DIN 18065 comprises specific requirements, e.g., concerning a stair's dimensions. Therefore, MBO's missing requirements are complemented by different standards but restrict possibilities of mere compliance checking regarding the MBO. To prevent confusion of the underlying validated compliance scenario by compounding different regulatory documents, detailed documentation must be conducted. The reason for missing specific requirements is again the fact that MBO only aims to set boundaries for federal law and is supplemented by different standards and regulations (PREIDEL, 2020).

### 4.3.2 Complexity

Transforming the information of a standard into machine-readable information for ACCC is beneficial to be able to identify the complexity of required process steps. For this purpose, the concept of complexity was introduced by Solihin and Eastman. 4 different categories can be distinguished:

**Tests with Explicit Input Data:** It is the lowest entity in the complexity hierarchy and tests for directly accessible attributes in the building model. Therefore, further computations are not required, but the accessed information can be compared to a specific value range or used in other equations. An example would be the information if a wall is external: this value can be retrieved via the PropertySet *Pset\_WallCommon* and the boolean property *IsExternal*.

**Tests with Derivative Information:** This represents the next step of complexity and applies to information that is not directly included in the building model. This data is temporarily generated in a particular process without complex calculations. A typical example is the distance calculation between two objects.

**Tests with Extending Data Structure:** Tests with extended data structure use information that is not included in the building model. External libraries are used due to their complex calculations methods and data structures to generate the required information. For example, the computation of escape routes might require particular libraries like graphs.

**Tests for Validation:** This category represents compliance checks relating to performance-based approaches. The central substance of the test does not refer to adherence to a particular claim, instead, it proves an overall solution. A common way of validation is the reference approach. It compares available solutions with the examined object. Examples for this category are guidelines for construction solutions that only represent assistance for particular scenarios and are not the only valid solution. These reference approaches are hard to formalize and, therefore, represent the highest complexity category (PREIDEL, 2020, SOLIHIN and EASTMAN, 2015).

## 4.4 Extraction Methods

The AEC sector is regulated by a large number of regulations on a national and international level. Additionally, BIM processes often come along with further information-related requirements in the form of, e.g., client requirements. The extraction procedure of information from these regulations and guidelines is often difficult and time-consuming and is complicated by different representation types. Automating this extraction process would drastically reduce the time consumption in a code compliance checking test's development phase. Furthermore, the resulting identification of relevant additional data and contents of a compliance test would ease the implementation process. In the last years, different approaches of information extraction were the subject of several research programs. For this thesis, the Mark Up Language RASE is outlined to identify and highlight essential elements of sentences. Additionally, the process and its different technical approaches of full automation text recognition and interpretation via Natural Language Processing (NLP) is reviewed.

### 4.4.1 Semantic Mark-Up RASE Methodology

The RASE method is based on a semantic concept and enables a translation of a norm's content into well-structured information. Therefore, it utilizes mark-up operators to identify relevant information which needs to be covered for a code compliance checking process. Every normative document postulates claims that have to be satisfied. These claims include one or several *requirements* which RASE identifies. Every claim refers to a

particular object or issue that RASE characterizes as the *applicability* object. The other two operators apply for *selections*, and identify alternatives, or *exceptions* and work by exclusion (HJELSETH and NISBET, 2011). All identified operators are marked via different colored RASE tags and facilitate a well structured and clear representation of the underlying content (see 4.1).

```
<R>Standard NS 11001-1, Clause: 5.2 Dimensioning an <a>access route</a> to a building
<R> The <a>access route</a> for <s>pedestrians</s><s>wheelchair users</s> shall <r>not
be steeper than 1:20</r>. <E>For <a>distances of less than 3 metres</a>, it may be steeper,
but <r>not more than 1:12</r>. </E> </R>
<R>The <a>access route</a> shall have <r>clear width of a minimum of 1,8 m</r> and
<r>obstacles shall be placed so that they do not reduce that width</r>. <r>Maximum cross
fall shall be 2 %.</r></R>
<R>The <a>access route</a> shall have <r>a horizontal landing at the start and end of the
incline</r>, plus <r>a horizontal landing for every 0,6 m of incline</r>. <r>The landing
shall be a minimum of 1,6 m deep.</r></R>
<R><r>Minimum clear height shall be 2,25 m</r> for the full width of the defined walking
zone of the entire <a>access route</a> including crossing points. </R></R>
```

Figure 4.1: Underlying RASE Mark-Up Tags Applied on Standard NS 11001-1 (HJELSETH and NISBET, 2011)

RASE is optimized for interpreting regulatory documents but must be able to handle variations in the text representation. Therefore, three principles are introduced to address the resulting semantic issues. The principle of *Translate* is applied if norms or regulations are expressed with clear metrics. In this case, direct insertion of RASE operators into the original text is possible. If the operators can not be directly applied, certain metrics are differently defined, or the underlying document is ill-drafted. The principle of *Transform* comes into place, and a reformulation of the text is required to enable an application of RASE on the reformulated text. In cases of strong generally formulated claims (*Transfer*), the identified unclear information is presented to the user for a manual interpretation by a professional (HJELSETH and NISBET, 2011).

The RASE mark up technique facilitates a well-structured text by highlighting relevant information. HUDECZEK, 2017 uses this methodology for automating information extraction. Additional tags extended the RASE tags to cover, e.g., references to pictures, tables, or other norms, or to identify specific parameters and values. This required extension outlined its limitations for an automated approach. In addition, RASE syntax is only applied to textual representations and comes with the loss of information regarding representation types like tables or pictures. The aforementioned extension of RASE tags to cover references of images or tables reduces the chance of overall information loss but does not improve information extraction processes. Nevertheless, this work aims to perform manual information extraction due to missing proper technical solutions. RASE can be manually utilized to understand the prevailing norm and regulation structure generally.

## 4.4.2 Natural Language Processing

**NLP** is a sub-field of computer science and linguistics and utilizes Artificial Intelligence (**AI**) to process written or spoken words via computers in a human-like manner. The term of natural language describes all written or spoken words by human beings. Since the 1950s, studies about the utilization of computers to understand and extract information from natural language are conducted. **NLP** deals with understanding and generation of natural language and speech recognition. The latter represents a more advanced approach due to additional difficulties resulting from spoken languages, like covering dialects (RESHAMWALA et al., 2013). The following outlines the approach of **NLP** to extract relevant information of regulatory texts or norms.

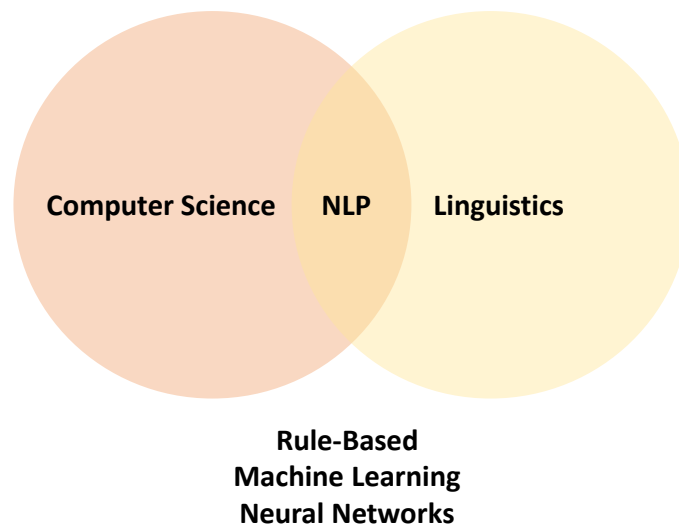


Figure 4.2: **NLP** Position in Computer Science and Linguistics, and its Possible Technical Implementations

**NLP** belongs to the broad field of **AI** applications and utilizes different technical approaches for the information extraction process. The first approach represents the oldest approach and uses manually coded rules to detect and categorize information. Newer approaches comprise Machine Learning (**ML**) or neural networks to analyze and extract natural language. **NLP** includes numerous fields of application, and its different technical approaches can match these varying requirements. A broad application field of texts covering the content of many domains, e.g., news texts, requires a more flexible approach which is mostly covered by **ML** or neural network solutions. These technical approaches require a large extend of information for their learning process but can learn to adapt to different information fields. Two main challenges of information extraction are the diverse vocabulary the software has to cover and understand, and the context and syntax ambiguity because of numerous different domains. ZHANG and EL-GOHARY, 2015 focus on the domain-specific information extraction regarding regulatory texts and processing of information to enable automated compliance checks via extracted data in the construction sector. Therefore, a rule-based approach based on syntactical and semantic features is utilized to increase its precision. The following comprises the methodology of a

domain-specific and rule-based approach and describes different methods and concepts which are frequently used in an information extraction process.

Domain-specific information extraction focuses on a particular domain to narrow the diverse vocabulary and reduce context and syntax ambiguity. Text analysis can be conducted concerning its syntax and semantics and requires a preprocessing step. The preparation of underlying texts can be processed via several different methods. The tokenization method splits the underlying text into units or called tokens. This process depends on the overall methodology, varying from word to subword or character tokenization. The overall tokenization process represents a preprocessing step to enable sentence splitting and POS tagging. The before-mentioned process of sentence splitting identifies every sentence of a text. Tokenization and sentence splitting utilize a particular delimiter to recognize required elements.

Furthermore, texts include words in various forms, like different tenses or plural forms. Morphological analysis help to identify the underlying word and map these to its lexical representation in the dictionary. This method is not essential for NLP but can enhance the identification of words and thus the underlying ontology concepts. An ontology is a data model that outlines a domain's knowledge and comprises its data and related relations. Therefore, an ontology is represented by classes and its relationships. The most common standards to create an ontology are Resource Description Language (RDF) and Ontology Web Language (OWL) and are easily extensible to cover more data and relationships. For example, an ontology regarding the construction sector covers numerous objects from this domain and its relations. Due to its specific domain, the underlying ontology has to cover less range of data which results in a higher interpretability and a enhanced preciseness of understanding domain specific texts.

As information representation, ZHANG and EL-GOHARY, 2015 utilizes a tuple format due to its straightforward manipulation and evaluation possibilities. Every object of this tuple format is a semantic information and represents an ontology concept, an ontology relation, a deontic operator indicator, like permission or prohibition, or a restriction. The latter represents a constraint regarding the semantic information element's definition. These identified semantic elements can then be classified by their complexity and flexibility, and a varying number of concepts and relations can be assigned. Thus, the first three types of semantic information are simple and rigid elements, and the last type, restrictions, represents a complex and flexible element.

Generating features to cover the text's content and meaning is an additional task besides the preprocessing of underlying texts. Therefore, syntactic features like Part-Of-Speech (POS) tagging, Phrase Structure Grammar (PSG) based phrasal tags and gazetter terms can be utilized in combination with semantic features of concepts and relations to identify and define patterns. POS tagging classifies each word or other tokens, like numbers or symbols, with its lexical and functional category. Table 4.1 outlines an example of possible tags to categorize identified tokens. The assigned POS tags can be

utilized for a structural phrase analysis like a noun phrase, verb phrase, or prepositional phrases, etc.

Table 4.1: Examples of POS Tags to Enable Classification of Tokens (ZHANG and EL-GOHARY, 2015)

POS Tag	Meaning
NN	Singular or Mass Noun
NNS	Plural Noun
NNP	Singular Proper Noun
JJ	Adjective
VBD	Past Tense Verb

PSG utilizes application-specific rules, developed on the basis of a random sample text, and is used to generate phrasal tags by identifying certain POS tags patterns. PSG represents a sentences' structure and facilitates the encoding of extensive sentence structures to reduce its complexity. PSG can be further reduced and results in Context-Free Grammar (CFG) to additionally decrease the number of patterns for the information extraction. In a study of ZHANG and EL-GOHARY, 2015 the implementation and utilization of PSG tags reduce the number of required patterns from 46 to 22 by more than 50%. Gazetteer lists comprise a set of specific entities and group these terms on one particular similarity. An example of a gazetteer list is the grouping of words like "no" and "not" in the "negation" gazetteer list. In addition, numerous gazetteer lists are available for different applications, like currency, countries, etc. These lists can be utilized as a feature for information extraction processes and increase computational efficiency by separating them from an overall ontology.

The overall information extraction process is realized by implementing rules to extract semantic information and rules that verify and resolve conflicts by defining a resolution strategy. The former are called information extraction rules and the latter conflict resolution rules. These rules utilize the before-generated features and concepts of underlying ontology to identify patterns and determine extractable parts. Conflict resolution rules address different occurring problems in the extraction process. For example, the overall number of particular present information elements in a sentence can exceed or fall below a certain number. Therefore, conflict resolution rules claim borders for individual information elements and are essential in information extraction processes. These conflict resolutions have to address specific scenarios regarding the underlying use case and have to be adjusted in the iterative development process of extraction rules. The development of rules for a rule-based approach is conducted by constructing identified patterns, possible relating features to select for underlying patterns and the overall semantic mapping. Construction of a set of identified patterns represents an iterative and empirical process by utilizing manual text analysis, constructing the overall pattern, testing, and validating results. The process of feature selection comprises the identification and selection of all present features in a pattern. Semantic mapping describes the transition of POS tags patterns and their related semantic instances. ZHANG and EL-GOHARY, 2015 states, a rule-based approach is favorable in cases of analyzing domain-specific texts due to

fewer ambiguity conflicts and, therefore, easier development of an ontology. Generally, rule-based NLP has better precision and recall results compared to machine learning but requires more human effort for its development of proper rules.

The complexity of information extraction regarding code compliance checking requires extracting event information, e.g., requirements or restrictions. Simple information extraction processes relate to named entity, attribute, and relation extraction. These categories utilize single or two related concepts, except for the event extraction, which requires numerous concepts. The term of deep NLP describes the aim to understand entire sentences and, therefore, enable extraction of all underlying information. Shallow NLP comprises only partial analysis of a sentence and capturing only specific types of information. The evaluation of information extraction results relies on a *golden standard*. *Golden standards* are manually examined by domain experts and represent the entirety of identifiable information of the underlying text (ZHANG and EL-GOHARY, 2015).

## 4.5 Summary

Norms and regulatory texts are common in economic sectors to guarantee high quality and clarify all stakeholders' rights and obligations. In Germany, regulations, standards, and guidelines are published by different institutions and vary in their status of legal character and application fields.

The MBO represents a regulatory text to ensure standardization of federal-state building regulations. The examined section 5 of MBO covers requirements concerning the fire protection of buildings and comprises claims regarding the escape ways and its associated building elements like stairs, stairwells, corridors, openings, and barriers of a building.

Regulatory texts and norms can be categorized regarding their type of representation and complexity to cover its requirements. Representation types comprise prescriptive or performance-based approaches. The former restricts or claims particular designs or materials, whereas performance-based approaches claim a level of quality. The latter facilitates new technical solutions because of not required admissions. MBO itself has a strong performance-based character due to its general approach standardizing federal law. Additional types of representation are tables and pictures utilized in many norms or other regulatory texts. The level of complexity refers to its required process steps of validating information. It outlines four different levels, comprising tests of simply querying data from a building model to information derivation from simple or complex computations.

To test for compliance with underlying norms and regulatory texts, extraction of information is required to generate computer-interpretable information. This work outlines the RASE mark-up methodology to enable the user to highlight and understand the essential structure of the digested contents by applying RASE tags. The NLP represents an AI based approach to process and extract information from texts automatically.

## Chapter 5

# Concept

The previous chapters outlined the fundamental understanding of BIM and its necessary constituent parts. Furthermore, the process of code compliance checking and its different approaches was reviewed, and finally, the classification of standards and automated information extraction methods were outlined. The following sections comprise a concept to identify and process relevant information of the fifth section of MBO. Additionally, the mapping process of additional data is described and overall characteristics of compliance checking procedures are reviewed.

### 5.1 RASE Mark Up Technique

Chapter 4 covered two variations of textual processing techniques. This thesis relies on a manual approach based on the before mentioned RASE mark-up methodology by [HJELSETH and NISBET, 2011] to structure and highlight all required information of a regulatory text. All regulatory texts contain several checks, described by at least one requirement, which are postulated by *shall* or *must*. The Applicability operator marks the related object of a requirement, which the Selection operator can further specify. A separate specification of applicability is the Exception to set certain exclusion of objects if required. The operators enclose the particular word or part of a sentence and are highlighted by different colors. The clear benefit of this technique is the explicit classification of textual segments which instantly emphasize the importance of certain sections. This reduces the amount of pure text to the essential, which is an advantage for straight continuous texts like the MBO.

Figure 5.1 shows an extraction of the § 33<sup>th</sup> paragraph of the MBO. In the first section, the utilization unit is the applied object enclosed with marked green operators. Additionally, the following red Selection operator highlights the specification if the utilization unit contains at least a single common room, two independent escape routes leading to open-air are required. This requirement is set apart by blue enclosing operators. In the next part, an Exception, marked by yellow operators, outlines a new Applicability, now referring to the escape route itself. Even manually performing the RASE mark-up technique ensures a better understanding of the standard's context and enhances the overall clarity of presented information for the user. The full RASE marked version of the MBO can be found in the Appendix A.



### § 33 Erster und zweiter Rettungsweg

(1) Für `</a>`Nutzungseinheiten`</a>` `</s>` mit mindestens einem Aufenthaltsraum`</s>` wie Wohnungen, Praxen, selbstständige Betriebsstätten `</r>` müssen in jedem Geschoss mindestens zwei voneinander unabhängige Rettungswege ins Freie vorhanden sein`</r>`;  
`</r>` `</e>``</a>` beide Rettungswege`</a>` dürfen jedoch innerhalb des Geschosses über denselben notwendigen Flur führen.`</e>``</r>`

(2) Für `</a>`Nutzungseinheiten`</a>` `</s>` nach Absatz 1`</s>`, `</s>` die nicht zu ebener Erde liegen`</s>`, `</r>` muss der erste Rettungsweg über eine notwendige Treppe führen`</r>`.  
Der `</a>`zweite Rettungsweg`</a>` `</r>` kann eine weitere notwendige Treppe`</r>` oder eine `</r>` mit Rettungsgeräten der Feuerwehr erreichbare Stelle`</r>` der Nutzungseinheit sein.  
`</e>` Ein `</a>`zweiter Rettungsweg`</a>` ist nicht erforderlich, wenn die `</r>`Rettung über einen sicher erreichbaren Treppenraum`</r>` möglich ist, in den Feuer und Rauch nicht eindringen können (Sicherheitstreppenraum).`</e>`

(3) Gebäude, deren `</a>`zweiter Rettungsweg`</a>` `</s>` über Rettungsgeräte der Feuerwehr führt`</s>` und bei denen die `</s>`Oberkante der Brüstung von zum Anleitern bestimmten Fenstern oder Stellen mehr als 8 m über der Geländeoberfläche`</s>` liegt, `</r>` dürfen nur errichtet werden, wenn die Feuerwehr über die erforderlichen Rettungsgeräte wie Hubrettungsfahrzeuge verfügt`</r>`.`</e>`  
Bei `</s>`Sonderbauten`</s>` ist der zweite Rettungsweg über Rettungsgeräte der Feuerwehr nur zulässig, wenn `</r>`keine Bedenken wegen der Personenrettung`</r>` bestehen.`</e>`

Figure 5.1: Mark-Up Technique Based on the RASE concept

The highlighted text of 5.1 includes all information necessary for a code compliance check. The Application of the first section is a utilization unit which can be represented as an *IfcSpace*. The spaces' identification can be guaranteed by a naming convention, classification, or additional mapped attributes to enable an automated process. Further, the Selection operator highlights the information of the existence of at least one common room. The implicit identification of spaces being a common room is not possible. Rooms could be identified via a naming convention and in combination with an underlying database to capture specific space types. However, the resulting database would be hard-coded, and the applied naming conventions have to match the implemented ones. This would result in a rigid workflow and could cause time delay. Therefore, an additional boolean attribute is more flexible and must be mapped to spaces to identify common and non-common rooms. The first requirement includes the term of escape ways in a story. Escape routes do not have a class representation in IFC like spaces but can be computed and represented by graphs. Therefore, knowledge regarding the different representation types in IFC and capability of underlying development environment are essential to be able to categorize highlighted information.

In many cases, it is helpful to extract the marked information in a table format. The user engages more often with the new information and ensures a better understanding of its context, which is crucial for a consistent ACCC process. Additionally, the table's content summarizes the received information clearly, which can be used for the check implementation later on. For example, in section (3), an exception for special buildings is

mentioned but not further specified, which can be easily completed in a table format. The table (Mark\_Up\_Table.xlsx) can be found in the Appendix A.

RASE mark-up operators help identify all essential information vastly. Still, detailed knowledge of the underlying data format and development environment is required to assure a flawless code compliance checking scenario implementation. Furthermore, unnecessary linked data in a building model must be prevented urgently due to possible occurring inconsistencies or interference with other stakeholders' information (BORRMANN et al., 2015). The following section covers all included fundamental data in IFC for ACCC to prevent needless mapped data.

## 5.2 IFC Data Coverage

This chapter comprises the core contents of IFC which build the basis for a reliable code compliance checking process. An understanding of common data delivered by a building model in IFC is necessary for an efficient mapping process of additional required data. Therefore, the spatial aggregation and its relating classes are represented, followed by an overview of components aggregation and possibilities in the open data scheme to assign additional information. The last section outlines the importance of the correct allocation of building elements to ensure a flawless workflow.

### 5.2.1 Three-Dimensional Aggregation

Compliance checking of the MBO requires detailed information about space-space and space-object relations. As mentioned in Figure 5.1, the first section (1) demands the existence of at least one common room in a utilization unit to require two independent escape ways. This semantic could be translated by additional information like room labels linking the common rooms to their relating utilization unit. However, additional mapped information must be reduced to a minimum and these relations can be computed by the standard spatial information of building models transferred via IFC.

First of all, the general principle of spatial aggregation in IFC has to be understood, before reducing the amount of additional data. As mentioned in Chapter 2, IFC is an object-oriented data format representing all occurring elements as discrete objects with specific attributes. For example, all spatial elements of a building project are represented by the class *IfcSpatialStructureElements* and its sub-classes *IfcSite*, *IfcBuilding*, *IfcBuildingStorey*, and *IfcSpace* ("buildingSMARTe", 2021). Its hierarchical aggregation is represented by the relation object *IfcRelAggregates*. It links the project and the site object, the building to its site, all building storeys to the building and all spaces (*RelatedElements*) enclosed by parenthesis to its relating building story (*RelatingElement*) as shown in Figure 5.2 and 5.3.

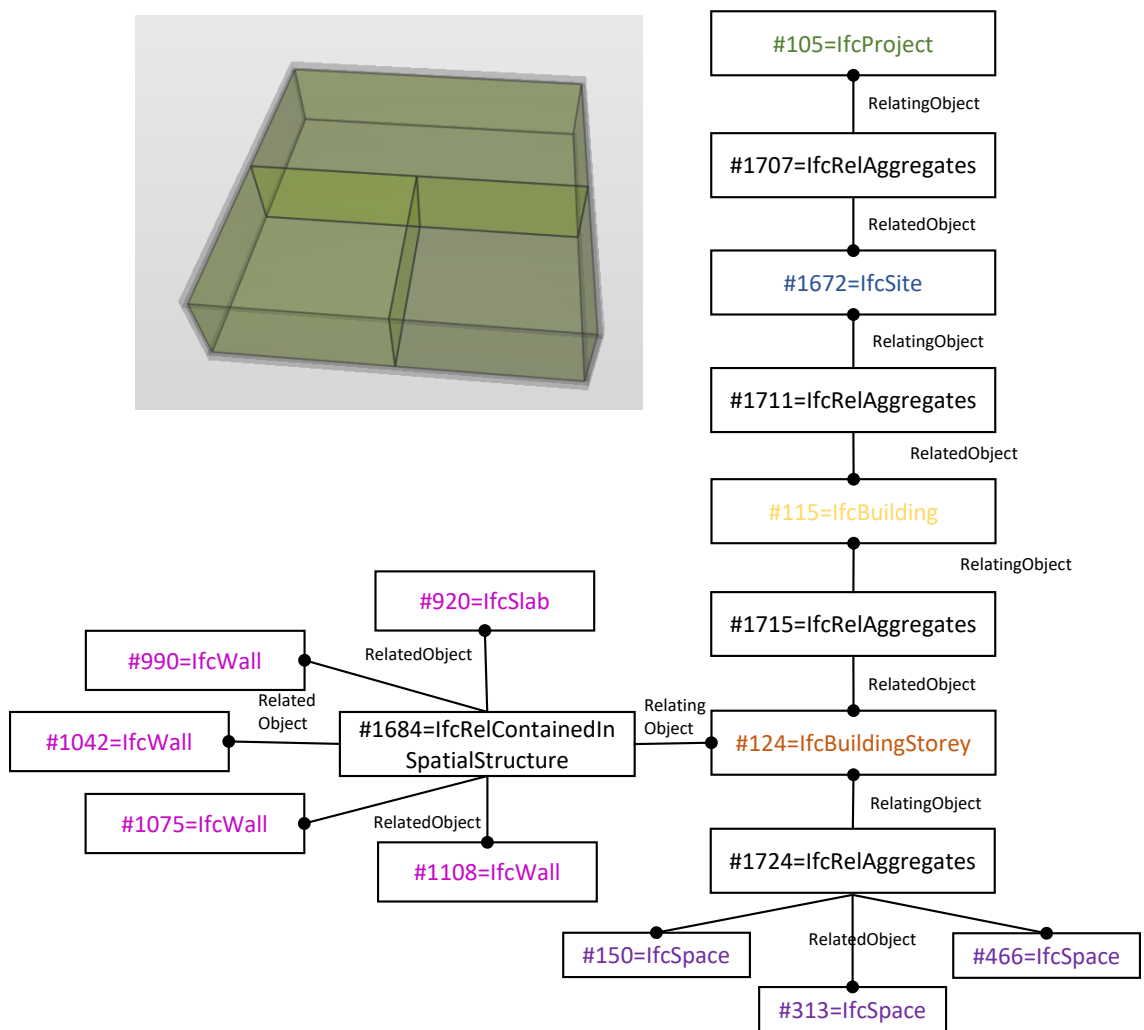
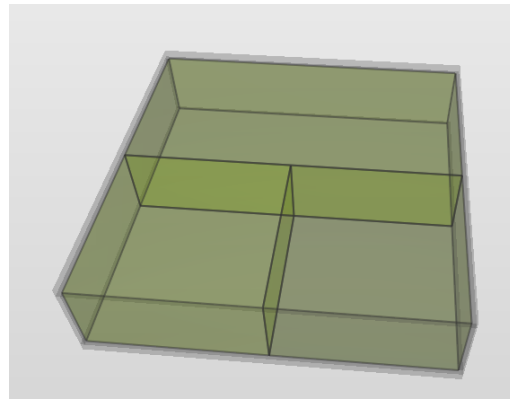


Figure 5.2: An Example of the Hierarchical Aggregation of Spatial Structure Elements and Assignment of Building Elements

The first four attributes are derived from the base class *IfcRoot* (see 2.2.2) and represents the **UID**, the *IfcOwnerHistory*, an optional name, and optional description. All objects' attributes are enclosed by parenthesis, separated by a comma, and always end with a semicolon. Continuing attributes differ from class to class but can be found on the **bSI** technical web page for each class and **IFC** version ("buildingSMART", 2021). This aggregation describes in detail its relations but doesn't reference any absolute or relative positioning of its elements in 3D space which is another mandatory information for **ACCC**.

```

#105= IFCPROJECT('3iIPdhi8D7zAYir0xdDagt',#41,'Project Number',,$,$,'Project
Name','Project Status',(#97),#92);

#1672= IFCSITE('3iIPdhi8D7zAYir0xdDagr',#41,'Default',,$,"#1671,$,$,.ELEMENT.,
(42,21,31,181945),(-71,-3,-24,-263305),0.,,$,$);

#115= IFCBUILDING('3iIPdhi8D7zAYir0xdDags',#41,",$,$,#32,$,$,.ELEMENT.,,$,$,#111);

#124= IFCBUILDINGSTOREY('3iIPdhi8D7zAYir0uOoOP8',#41,'Level 1',,$,$,#122,$,'Level
1',.ELEMENT.,0.);

#150= IFCSPACE('3oGUzJFLD85gDtudg9VrrN',#41,'1',,$,$,#133,#146,'Corridor',.ELEMENT.,
.INTERNAL.,$);

#313= IFCSPACE('31sRVdxoTEIhSFYtHosqt',#41,'1',,$,$,#280,#311,'Nutzungseinheit',
.ELEMENT.,.INTERNAL.,$);

#466= IFCSPACE('31sRVdxoTEIhSFYtHosqqQ',#41,'2',,$,$,#434,#464,'Aufenthaltsraum',
.ELEMENT.,.INTERNAL.,$);

#1707= IFCRELAGGREGATES('1qU9WVPLH1hhGls1_OPZdV',#41,$,$,#105,(#1672));

#1711= IFCRELAGGREGATES('0ROFvV0VL8YRr2p9_rt61C',#41,$,$,#1672,(#115));

#1715= IFCRELAGGREGATES('1vK3KfqgSHqv5Y00A6FnIY',#41,$,$,#124,(#150,#313,#466));

#1724= IFCRELAGGREGATES('3ioAY2VOL1QOEtMGyzl4Rh',#41,$,$,#115,(#124));

```

Figure 5.3: Aggregation of Spatial Structure Elements

Positioning of components in IFC, except for *IfcSite*, is secured by relative positioning via local coordinate systems, the *IfcLocalPlacement* class. *IfcLocalPlacement* is always linked in the relating object's attributes and contains as the first attribute another *IfcLocalPlacement* entity. This optional attribute provides the related parent coordinate system. If it's not set, the object is placed absolutely in the global coordinate system. The second attribute is the *RelativePlacement* and contains the transformation from the parent coordinate system to the local placement. *RelativePlacement* uses the *IfcPlacement* parent class and its sub-classes *IfcAxisPlacement*, *IfcAxis2Placement2D* and *IfcAxis2Placement3D*.

```

#6= IFCCARTESIANPOINT((0.,0.,0.));

#1670= IFCAXIS2PLACEMENT3D(#6,$,$);

#1671= IFCLOCALPLACEMENT($,#1670);

#1672=IFCSITE('3iIPdhi8D7zAYir0xdDagr',#41,'Default',,$,"#1671,$,$,.ELEMENT.,(42,21,31
,181945),(-71,-3,-24,-263305),0.,,$,$);

```

Figure 5.4: Absolute Positioning of *IfcSite*

Figure 5.4 depicts the particular case of the *IfcSite* entity and its placement in the global coordinate system. *IfcLocalPlacement* attribute's *RelativePlacement* is missing and substituted with \$ and, therefore, is not placed relatively to another component, but in the global coordinate system's origin. Next to the spatial building entity relations and their geometric positioning, the allocation of the remaining components, like walls, columns,

etc., to their relating spatial elements plays an important role for [ACCC](#). This eases the overall process of finding space-object relations.

A standard assignment of elements to its containing spatial structures in [IFC](#) is realized by the *IfcRelContainedInSpatialStructure* class. As mentioned above, spatial structures are *IfcSite*, *IfcBuilding*, *IfcBuildingStorey* and *IfcSpace* and its relating building elements are assigned. *IfcRelContainedInSpatialStructure* contains the common four attributes of *IfcRoot* and holds as its fifth attribute a set of *RelatedElements*, representing the assigned building elements, enclosed in parenthesis. The last attribute covers the *RelatingElement*, like in [Figure 5.5](#) an *IfcBuildingStorey*.

```
#124= IFCBUILDINGSTOREY('3iIPdhi8D7zAYir0uOoOP8',#41,'Level 1',$,$,#122,$,'Level 1',.ELEMENT,.0.);

#920= IFCSLAB('3oGUzJFLD85gDtudg9VrFm',#41,'Floor:Generic Floor - 400mm:201980',$,'Floor:Generic Floor - 400mm',#898,#918,'201980',.FLOOR.);

#990= IFCWALL ('3oGUzJFLD85gDtudg9Vr83',#41,'Basic Wall:Generic - 200mm:201999',$,'Basic Wall:Generic - 200mm:249',#963,#988,'201999',.NOTDEFINED.);

#1042= IFCWALL ('3oGUzJFLD85gDtudg9Vr8S',#41,'Basic Wall:Generic - 200mm:202000',$,'Basic Wall:Generic - 200mm:249',#1022,#1040,'202000',.NOTDEFINED.);

#1075= IFCWALL ('3oGUzJFLD85gDtudg9Vr8T',#41,'Basic Wall:Generic - 200mm:202001',$,'Basic Wall:Generic - 200mm:249',#1055,#1073,'202001',.NOTDEFINED.);

#1108= IFCWALL ('3oGUzJFLD85gDtudg9Vr8U',#41,'Basic Wall:Generic - 200mm:202002',$,'Basic Wall:Generic - 200mm:249',#1088,#1106,'202002',.NOTDEFINED.);

#1684=
IFCRELCONTAINEDINSPATIALSTRUCTURE('1vK3KfqgSHqv5Y0066FnIY',#41,$,$,(#920,#990,#1042,#1075,#1108),#124);
```

Figure 5.5: Linking of Building Components to Spatial Elements

An additional possibility to receive advanced spaces-object relation can be achieved by changing the [IFC](#)-export settings. If *Space boundaries* are selected, the *IfcRelSpaceBoundary* class is added to the exported [IFC](#) file. It inherits, like all sub-entities from *IfcRoot*, an [UID](#), a relating *IfcOwnerHistory*, optional name, and description. The fifth attribute references the bounded space, followed by the building element that represents this boundary. Physical boundaries can be, e.g., *IfcWall*, *IfcSlab*, *IfcWindow*, and *IfcDoor*. Spaces can as well be limited in their expansion by virtual *Space Separators*. If the *Space boundary* represents a virtual border, the sixth attribute is empty and filled with \$. A further feature is the representation and calculation of occupied space area relating to surrounding *Space boundaries*. The seventh attribute passes the calculated area information whereby the overall area of a space can be computed and every *Space boundary* with its contribution. The last two attributes hold *enumerations* about whether it's a virtual

or physical border and internal or external. Figure 5.6 shows examples for a virtual and physical *Space boundary*.

```
#861=  
IFCSpace('31sRVdxoTEIhSFYtHosqrS',#42,'5',,$,$,#847,#858,'Space',.ELEMENT.,.SPACE.,$);  
  
#970= IFCSLAB('3oGUzJFLD85gDtudg9VrFm',#42,'Floor:Generic Floor -  
400mm:201980',,$,'Floor:Generic Floor - 400mm',#933,#966,'201980',.FLOOR.);  
  
#909=  
IFCRELSPACEBOUNDARY('1aqQfG6HP99RF1xpAIB1oV',#42,'1stLevel',,$,#861,$,#908,  
.VIRTUAL.,.INTERNAL.);  
  
#2295=  
IFCRELSPACEBOUNDARY('3tDiD7U1FsOjYJIWHOaQm',#42,'1stLevel',,$,#861,#970,#878,  
.PHYSICAL.,INTERNAL.);
```

Figure 5.6: Virtual and Physical Space Boundary

In terms of **ACCC**, the chosen level of *Space Boundaries* is secondary most of the time. *Level 2* gives additional information on whether a building element or space is on the other side. This information could as well be used for checking mechanisms but is mostly used for thermal computations (“buildingSMARTg”, 2021). *Space boundaries* are useful if space’s area must be assigned to particular building elements, e.g., space occupied by windows. For more general space information the *Ifc Quantity Set Qto\_SpaceBaseQuantities* can be used instead.

**IFC** has a strong representation of three-dimensional information, which can be utilized in code compliance checking procedures. Nevertheless, this information relies on the modeling process and has to be determined and agreed on in every project. For example, a digital building model can be modeled without proper building story assignment, leading to significant problems for hard-coded compliance tests if required. Therefore, these requirements must be addressed in a test’s documentation and a project’s exchange requirements.

## 5.2.2 Building Element Aggregation

The in 5.2.1 mentioned *IfcRelAggregates* class is used for object composition and decomposition as well. Certain building elements can be exported to **IFC** as an assembly of parts, representing the building element as an aggregation or as a single entity, directly including an object’s representation. As a composition of building elements, every single part holds its representation, and the whole building element is represented by this assembly, as shown in Figure 5.7 (“buildingSMARTi”, 2021).

```

#145= IFCLOCALPLACEMENT(#134,#144);

#147= IFCSTAIR('0Yj0wT8r6k8hjf$fbth9h',#42,'AssembledStair:Stair:209667',$,'Assembled
Stair:180mm max riser 275mm tread:159374',#145,$,'209667',.NOTDEFINED.);

#4871=IFCRELAGGREGATES('0Yj0wcT8r6k8hjf$bbth9h',#42,$,$,#147,(#1033,#1067,#1098,
#3014,#4860));

#1033= IFCSTAIRFLIGHT('0Yj0wcT8r6k8hjf$fbth9r',#42,'Assembled Stair:Stair:209667 Run
1',$,'Assembled Stair:180mm max riser 275mm tread:159374',
#162,#1028,'209693',23,22,0.57058085130663,0.902230971128609,.NOTDEFINED.);

#1067= IFCMEMBER('0Yj0wcT8r6k8hjf$fbth9t',#42,'Assembled Stair:Stair:209667 Stringer
1',$,'Assembled Stair:180mm max riser 275mm tread:159374',
#1037,#1061,'209695',.STRINGER.);

#1098= IFCMEMBER('0Yj0wcT8r6k8hjf$fbth98',#42,'Assembled Stair:Stair:209667 Stringer
2',$,'Assembled Stair:180mm max riser 275mm tread:159374',
#1071,#1095,'209696',.STRINGER.);

#3014= IFCRAILING('0Yj0wcT8r6k8hjf$fbth90',#42,'Railing:900mm Pipe:209704',$,
'Railing:900mm Pipe',#1181,#3011,'209704',.NOTDEFINED.);

#4860= IFCRAILING('0Yj0wcT8r6k8hjf$fbth94',#42,'Railing:900mm Pipe:209708',$,
'Railing:900mm Pipe',#3030,#4857,'209708',.NOTDEFINED.);

#4857= IFCPRODUCTDEFINITIONSHAPE($,$,(#4853));

```

Figure 5.7: Object's Shape Representation of *IfcStair* as an Assembly of Building Elements

The *IfcStair* entity holds an *IfcLocalPlacement* attribute followed by \$. This missing attribute states the *IfcProductDefinitionShape*, enclosed in parentheses and therefore, represents a set of possible *IfcShapeRepresentations*. A missing *IfcProductDefinitionShape* indicates an assembled *IfcStair* object; thus, all aggregated elements hold their shape representation. The highlighted attributes in the stair's sub-elements hold the *IfcProductDefinitionShape* attribute, outlined for the last element in line #4857, *IfcRailing*.

Regarding ACCC, an export of specific categories of building elements must be considered depending on the required level of detail in respect of claims made by regulations and standards. As mentioned in chapter 3, hard-coded tests rely on strict abundance by correct data-object assignment. Therefore, all required elements must be assigned and exported for a flawless and accurate compliance process. These export requirements must be outlined and determined in ER and IDM process. The following covers all assembled standard building elements and their export requirements optimized for ACCC regarding the MBO in this thesis.

## IfcStair

A stair can be an assembly entity or a single entity. In this thesis, stairs are exported as an aggregation of building elements. A stair is assembled by *IfcStairFlight*, representing risers and treads, *IfcRailing*, *IfcSlab (Landing)*, and *IfcMember* representing parts like its bearing structure. [bSI](#) states to assign *IfcMember* to entities of *IfcBeam* or *IfcColumn* if it can be expressed more specifically (“buildingSMART”, 2021). These concerns must be supervised in every particular project in consultation with structural engineers, but concerning this thesis, no further specifications will be made concerning *IfcMembers*. An assembly entity for stairs, checking for the fifth section of the [MBO](#), is beneficial, e.g., due to detailed demands for the worst fire resistance stair’s components. Therefore, all fire resistance properties are just assigned to its relating building element and the user doesn’t have to identify the decisive fire resistance by himself.

## IfcRamp

[IFC](#) enables aggregation of ramps composed of a ramp flight and landings, which are represented by slabs. These slabs can contain an enumeration, like *LANDING*, in its *PredefinedType* attribute to define its specific slab type. Like for stairs mentioned above, railings are as well linked to the *IfcRamp* entity by *IfcRelAggregates*. Regarding this thesis, it is an advantage to attach all railings to its relating stair or ramp entities because [MBO](#) claims their existence.

## IfcCurtainWall

Curtain walls are complex building elements, and therefore its sub-elements can hold vital information for [ACCC](#). *IfcCurtainWalls* are composed of *IfcPlates* representing the glazing panels and the mullions by *IfcMembers*. As already mentioned above, for *IfcStair*, the specific assignment of *IfcMember* entities to *IfcBeam* or *IfcColumn* must be stated in every project but won’t be further specified in this thesis. Additionally, panels can be replaced by *IfcDoors* or *IfcWindows* and thus are part of the aggregation. This assignment of crucial escape way components strengthens the benefit of modeling and exporting curtain walls as assembly entities for compliance checking processes.

## IfcRoof

The architectural shape of roofs are numerous and can be represented by a single roof entity. *IfcRoofs* are aggregated by *IfcSlabs* and complemented by structural components like *IfcBeam* or voided and filled by building elements like *IfcWindow*. Detailed access to the roof’s sub-elements is not required due to missing claims concerning roofs and their



sub-elements in the [MBO](#), but in this thesis, the *IfcRoof* will be exported as an aggregated entity.

The specific building element aggregation outlined above represents an additional exchange requirement concerning this thesis and the compliance checking scenario for the fifth section of [MBO](#). Therefore, the underlying building elements are modeled and exported as outlined and are validated in the implemented compliance procedure.

The following concepts of additional semantic assignment in [IFC](#) do not directly apply to this thesis and should only be used if all participants agreed on the precise usage. Nevertheless, they are presented for reasons of completeness of [IFC](#) classes.

### 5.2.3 Additional Semantic Assignment

#### **IfcClassification**

[IFC](#) provides the functionality to array objects in a class or category by their principal purpose or characteristics. *IfcClassifications* build a hierarchical scheme and establish the functionality to apply taxonomies. *IfcClassifications* hold detailed information about the used classification systems like *OmniClass*, *UniFormat*, and many more. *IfcClassificationReferences* relate to its classification system (*IfcClassification*) and provide the classification information for specific building components. The relation object *IfcRelAssociatesClassification* connects a set of appropriate components and the belonging *IfcClassificationReference* as depicted in [Figure 5.8](#) (“buildingSMARTk”, 2021).

```
#314= IFCWALL('0Yj0wcT8r6k8hjf$fbthKy',#42,'Basic Wall:Interior - 138mm Partition (1-hr):210004',$,'Basic Wall:Interior',#285,#310,'210004',.NOTDEFINED.);
```

```
#431= IFCWALL('0Yj0wcT8r6k8hjf$fbthK_',#42,'Basic Wall:Interior - 138mm Partition (1-hr):210006',$,'Basic Wall:Interior',#409,#427,'210006',.NOTDEFINED.);
```

```
#578= IFCCLASSIFICATION('User','2020',$,'UserDefined',$,$,$);
```

```
#580= IFCCLASSIFICATIONREFERENCE($,'Shear Wall',$,#578,$,$);
```

```
#581= IFCREASSOCIATESCLASSIFICATION('pVZN3mirRru6najDa29G+g',#42,'Wall Classification',$,($,#314,#431),#580);
```

Figure 5.8: Classification of Shear Walls with a User Defined Classification System

Classifications are a widespread approach in a [BIM](#) workflow to enrich semantics. This functionality is helpful for [ACCC](#) in cases of allocating building components to a particular category containing various sub-elements. An example would be the use of rooms and categorization regarding its common room status. In this thesis, only property sets (see [2.2.3](#)) are used for data mapping to guarantee high clarity for the user and all stakeholders. Nevertheless, if code compliance checking process is well implemented and

enough experience was gained, a separation of mapped data in terms of the used data assignment methodology can be taken into consideration.

## IfcGroup

The aggregation of building elements mentioned above is a standard solution in IFC to provide an enhanced semantic to specific composite components. Modeling tools, like Revit, provide another functionality to represent relationships between building components by grouping them. In Revit, *Groups* can be used to allocate particular modeled building elements that are often used in the same aggregation. Using these groups allows a much faster and more efficient modeling process. This grouping semantic is as well integrated into the IFC data scheme, as shown in Figure 5.9.

```
#186= IFCWALL('0Yj0wcT8r6k8hjf$fbthA8',#42,'Basic Wall:Interior - 138mm Partition (1-hr):209888',$,'Basic Wall:Interior - 138mm Partition (1-hr):220',#147,#180,'209888',.NOTDEFINED.);  
#306= IFCWALL('0Yj0wcT8r6k8hjf$fbthLY',#42,'Basic Wall:Interior - 138mm Partition (1-hr):209930',$,'Basic Wall:Interior - 138mm Partition (1-hr):220',#284,#302,'209930',.NOTDEFINED.);  
#318= IFCGROUP('0Yj0wcT8r6k8hjf$fbthL1',#42,'Model Group:Test:209961',$,'Model Group:Test');  
#320= IFCREASSIGNSTOGROUP('0Yj0wcT8r6k8hjf$fbthL1',#42,$,$,(#186,#306),$,#318);
```

Figure 5.9: Grouping of two Walls in a *Test* Group

The *IfcGroup* class represents the actual grouping entity which is linked by the relation object *IfcRelAssignsToGroup* with the included set of *IfcWalls*. In general, bSI designed *IfcGroups* to nest "products, processes, controls, resources, actors or other groups, ..." ("buildingSMARTj", 2021) without dependencies or hierarchical meaning. Objects can be linked to several groups simultaneously, and groups can be assigned to further objects like processes, resources, and control entities via relationship classes like *IfcRelAssignsToProcess*, *IfcRelAssignsToResource*, and *IfcRelAssignsToControl* ("buildingSMARTj", 2021). Grouping via *IfcGroups* could be used in ACCC for additional relation semantic of spatially separated components. This should only be taken into consideration if all project's stakeholders agree on this approach and guarantees a significant improvement in clarity and saving of labor.

### 5.2.4 Building Element Allocation

Modeling tools provide a variety of standard building element categories to model, e.g., walls and slabs. These categories already hold numerous important element-specific properties and functionalities. For example, walls are automatically voided and filled with doors and windows. In IFC, *IfcRelFillsElement* and *IfcRelVoidsElement* facilitate this functionality by linking voiding and filling elements and reducing the computation effort in compliance checking processes drastically.

The appearance of specific building categories can be extended in Revit by the importation of families. These families represent user-defined components that have to be assigned to its relating Revit category. Later on, this assignment ensures the correct allocation to building element type in IFC. ACCC depends on delivered information of a building model, like correct type assignment of building elements. In compliance checking tests, this type of information is often used to sort components and test particular attributes. An incorrect allocation can prevent the assignment of information regarding particular building components and result in undetected errors.

The AEC sector has a large variety of building elements, and many are not covered by standard categories of IFC, e.g., ladders (see Figure 5.10). Hence, bSI provides the class *IfcBuildingElementProxy* with the same functionality as subtypes of *IfcBuildingElement* but does not represent a particular type of building components.

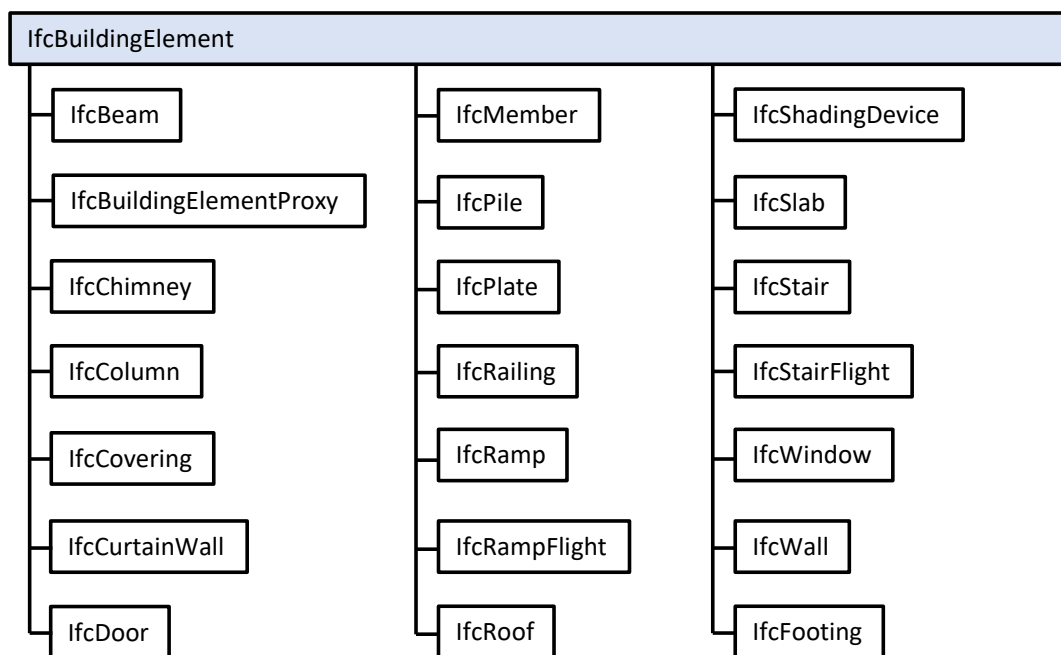


Figure 5.10: Entity Inheritance of *IfcBuildingElement* and its Sub-Classes (“buildingSMART”, 2021)

Building elements that can not be classified into one of *IfcBuildingElement*’s sub-classes can be assigned to the *IfcBuildingElementProxy*. This class provides the same functionalities as the other subtypes, but it is missing attributes providing meaning for a particular building element. Table 5.1 represents an additional part of the exchange requirements to check for compliance with the fifth section of MBO and depicts the underlying building element allocation regarding its IFC element type.

Table 5.1: Building Elements and relating Element Types in IFC

Building Element	IFC Element Type	Building Element	IFC Element Type
(Glass-)Door	IfcDoor	Window	IfcWindow
Railing	IfcRailing	Stair Steps	IfcStairFlight
Stair	IfcStair	Stair Elements	IfcMember
Ramp	IfcRamp	Ramp Flight	IfcRampFlight
Wall	IfcWall	Curtain Wall	IfcCurtainWall
Beam	IfcBeam	Column	IfcColumn
Floor	IfcSlab	Withdrawal	IfcB.El.Prox
Suspended Ceiling	IfcCovering	Area	IfcSpace
Escalator	IfcTransp.Elem.	Ladder	IfcB.El.Prox
Light Source	IfcLightFixture	Glass Cover (horiz.)	IfcRoof
Room	IfcSpace	Utilization Unit	IfcSpace
Opening	IfcOpening	Roof	IfcRoof
Skylight (door)	IfcWindow	Side Part Door	IfcWindow

The table above does not contain any requirements for coverings, plaster, and insulation which can be represented by *IfcCovering* and specified by its attribute *PredefinedType*. These coverings are part of multilayered components like walls, columns, or beams. The legal text claims, e.g., the fire resistance of surfaces, plaster and insulation, to match particular requirements. In this thesis, multilayered walls are exported as solid walls because they represent the most general approach and cover a low level of detail of even early project phases. The resulting fire resistance or behavior of walls, columns, and beams is realized by the most decisive element of this compound. The user must incorporate this issue in the mapping process. An advantage despite the additional labor during the mapping procedure is the applicability of resulting compliance tests in earlier project stages. Therefore, a less detailed model can be checked for compliance, and planning errors can be detected in early project phases. A part of the outlined building elements is utilized in the test building model and validated in the implemented compliance checking procedure.

### 5.3 Data Extraction and Mapping

Chapter 5.2 outlined various classes in the IFC scheme and how these standard semantics can be arranged and used for ACCC. The structured and prepared information of regulations and standards (see 5.1) have to be passed to all relating building elements via property sets. The following sections introduce the different levels of granularity and their impact on the information extraction process. In addition, the overall mapping process with all additional information of the MBO is presented and reviewed, and different code compliance checking approaches are outlined and discussed.

### 5.3.1 Granularity and Complexity

ACCC requires computer-readable information to test compliance with specific standards and regulations. The diversity of representation types (see 4.3) often provide different levels of descriptive precision and, therefore, lead to variable approaches of handling this information which will be outlined in the following section.

In general, norms claim certain issues and their abundance. The existence of a claim is essential to translate information into computer interpretable data for compliance checking (see 5.11). Hence, structuring information as shown in 5.1 outlines what information is demanded and might be checked.

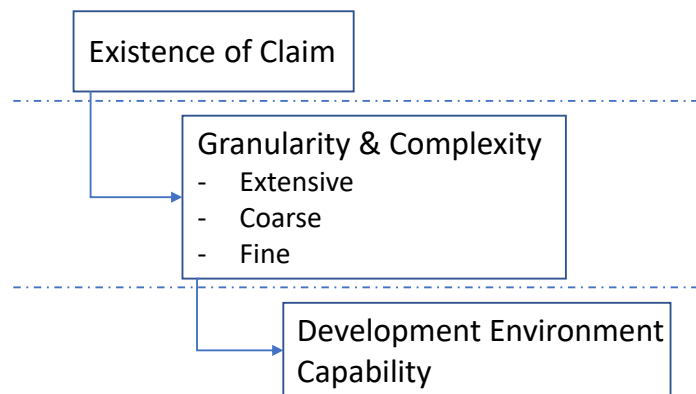


Figure 5.11: Three Fundamental Element Hierarchy for Information Translation into Computer Readable Data

A claim's granularity and complexity build the second hierarchical element. Granularity delivers information about semantic preciseness (BITTNER and SMITH, 2002). The finer its granularity, the easier it is to map as explicit data. Complex demands in regulations and standards must be disintegrated into simple, precise data that can be mapped to building components. Via these explicit input data, tests of higher complexity with extended data structure or even tests for validation of certain circumstances can be realized.

§35 Notwendige Treppenräume, Ausgänge

*1) Notwendige Treppen sind ohne eigenen Treppenraum zulässig ... als Außentreppe, wenn ihre Nutzung ausreichend sicher ist und im Brandfall nicht gefährdet werden kann.*

The MBO claims stairs, which are part of an escape route, do not have to be inside of a stairwell if it is an external stair and its use is sufficiently safe and not endangered in case of fire. The granularity is extensive and does not set any boundaries regarding the sufficient safety of external stairs. Therefore, further specifications are required to check for compliance automatically, e.g., incorporating the user and handing over unclear situations. The MBO is complemented by federal laws in a much more detailed version. Checking for compliance with the MBO can often be advantageous because of its uniformity for all federal states. Parts of regulations with an extensive granularity

should only be compounded with other complementing norms combined with detailed documentation to clarify the overall checking procedure. In a compliance checking process, it is always recommended to separate contents of different norms and test each claim individually. This distinct separation ensures well-arranged issue management.

#### §35 Notwendige Treppenräume, Ausgänge

*2) Sind mehrere notwendige Treppenräume erforderlich, müssen sie so verteilt sein, dass sie möglichst entgegengesetzt liegen und dass die Rettungswege möglichst kurz sind.*

The second section of §35 demands several stairwells to be well distributed and positioned in opposite directions. Additionally, its relating escape routes are supposed to be as short as possible. A coarse granularity combined with complex problems like this spatial and geometric question of stairwells distribution in buildings. Claims of this granularity are generally quantifiable but depend on the overall developer skills and the provided technical possibilities of the development environment. Requirements differ in their granularity and complexity, and the former represents a dynamic principle, so a uniform approach is not applicable. It always depends on the user's valuation. The process mentioned above of passing specific issues directly to the user for a detailed rating of the problem fits many cases with coarse granularity. If issues are describable via certain value boundaries or similar borders, default thresholds or input of user values can be utilized.

#### §36 Notwendige Flure, offene Gänge

*2) Notwendige Flure müssen so breit sein, dass sie für den größten zu erwartenden Verkehr ausreichen.*

This extraction of the fifth section of **MBO** corridors' width must match the maximum occurring traffic. The granularity is fine, but the claim is missing a boundary value. Fine granularity does not require the user's assessment and is solvable via thresholds or user inputs. Especially thresholds strengthen the black-box character of hard-coded code compliance checks because the user can not see the underlying thresholds. Inevitable thresholds require detailed test documentation why a implementation of user input is preferable.

Fine granularity and boundary values guarantee a quantifiable problem. The granularity's acuteness enables the user to identify all affected and involved building elements. Additionally, detailed relating information can be mapped and often complemented with boundary values. For example, the following extraction of § 34 regarding stairs outlines the requirements of a stair's bearing structure. Depending on the particular building class, the bearing element must match a specific fire resistance and behavior. All addressed building components and their additional attributes are specified and facilitate a clear identification for the mapping process.

§34 Treppen

*4) Die tragenden TeilerotwendigerTreppen müssen*

*1. in Gebäuden der Gebäudeklasse 5 feuerhemmend und aus nichtbrennbaren Baustoffen*

*2. in Gebäuden der Gebäudeklasse 4 aus nichtbrennbaren Baustoffen*

*3. in Gebäuden der Gebäudeklasse 3 aus nichtbrennbaren Baustoffen oder feuerhemmend*

The existence of a claim and its granularity define whether a circumstance is generally testable for compliance. The last element represents a software's or development environment's capability to test for the underlying issue. Therefore, the developer requires detailed knowledge about the used software combined with extensive know-how in the particular [AEC](#) field relating norms and regulations and the overall data coverage of [IFC](#).

### **5.3.2 Further Analysis and Strategies**

A claim's granularity specifies whether its comprised information is reducible to simple explicit data or requires further workarounds. In this section, additional assessment approaches to handle and review information extracted from standards are outlined. The outlined procedures enable users to manage various information types and make necessary decisions in an efficient extraction and mapping process.

#### **General Claims**

§33 (3) states that the second escape route via a window can only be realized if the fire brigade has the necessary life-saving equipment. The requirement's granularity is sufficient to map this information directly, but fire brigade objects are not present in [IFC](#). Hence, an integrated check-box in the interface ensures the availability of all required data without identifying concerned objects.

§33 Erster und zweiter Rettungsweg

*(3) ..., wenn die Feuerwehr über die erforderlichen Rettungsgeräte wie Hubrettungsfahrzeuge verfügt.*

## Implicit Semantics and Modelling Standards

One of [AEC](#) sector's characteristics is unique projects and resulting high complexity. Therefore, developing algorithms capable of handling a multitude of different scenarios (high level of generalization) requires a lot of coding knowledge and a reliable database.

§35 Notwendige Treppenräume, Ausgänge

*(4) Der obere Abschluss notwendiger Treppenräume muss als raumabschließendes Bauteil die Feuerwiderstandsfähigkeit der Decken des Gebäudes haben; dies gilt nicht, wenn der obere Abschluss das Dach ist und die Treppenraumwände bis unter die Dachhaut reichen.*

The first requirement of §35 (4) claims matching fire resistance of a stairwell's top closure (ceiling or roof) and the building's ceilings (*IfcSlab*). Therefore, all concerned objects of a stairwell can be accessed by the underlying three-dimensional aggregation in [IFC](#) (see [5.2.1](#)) and the highest building elements of either *IfcSlab* or *IfcRoof* can be accessed and evaluated regarding its fire resistance. Strict modeling standards can guarantee the exact stairwell's height to validate the top closure. Hence, the implicit derivation of the top closure can be implicitly derived, and additional mapping is averted.

The modeling standards mentioned above establish exact requirements in the modeling process and facilitate reliable core information, enabling replicability for underlying compliance scenarios. A modeling requirement would be the claim for the existence of spaces and their detailed modeling properties, e.g., space must expand to its bordering building components. These exact requirements should be included by the underlying test documentations and distributed to all affected stakeholders.

Besides the definition and implementation of modeling standards, the algorithms used to derive implicit information have to be evaluated for their steadiness regarding many occurring scenarios. Undetected errors lead to fatal mistakes in the issue management and result in time loss and additional costs. For this reason, the utilization of complex algorithms producing implicitly derived data is only applicable for scenarios with manageable complexity. Furthermore, §35 (4) includes detailed requirements concerning the stairwell walls reaching the roof membrane. Highly detailed modeling requirements are challenging to meet and, therefore, delivered as additional information (*ReachingRoof*) mapped to relating walls.

## Method Limitations and User's Incorporation

An insufficient granularity (see [5.3.1](#)) can be handled via incorporating the check's user and using his valuation of present problems. Despite an imprecise description of stated claims in a norm or regulation, the current limitations of [ACCC](#) have to be known and well understood by the user to apply sufficient methods preventing undetected errors. So far,



the following situations might appear in a code compliance checking process and require the human ability to make accurate decisions.

Incorporating users resembles the approach of serving concerned elements "on a tray" to reduce error rates and amount of work simultaneously. The goal of fully automated tests is not realizable in every scenario, but user incorporation can complement the current issue management process. An example is the missing Level of Detail (LoD) (not LOD). Although IFC covers a large number of classes to represent building elements of various types, correlations like free space in a stair between its railing and steps (§38 (1) 6.) are undetectable via algorithms. Additionally, reliable solutions are hindered by the plurality of design possibilities, and therefore, a domain expert's opinion can provide a remedy.

Furthermore, the representation of specific relations between coherent building elements like windows and its correlating device to open it (§35 (8)) is not possible in the current state (IFC 4). Additional data could solve this missing semantic, but in this case, the approach should be avoided due to the rapidly rising amount of data. Directly involving and presenting affected objects to the user prevents a large amount of additionally required data and results in active user involvement. Another example in MBO is §37 (1) that requires changes in the arrangement of its environment to check its validity. It claims the presence of elevators, mounts, or bars if windows can not be cleaned up from the ground floor, inside of buildings, or balconies. These constantly changing situations require a valuation of its subprocesses and are not evaluable in an ACCC process, which only represents a particular project phase. Regarding this scenario, IFC provides functionalities of assigning building elements to certain project phases to simulate processes.

The different approaches outlined above cover various scenarios to identify essential information and handle insufficient stated requirements or technical realization possibilities. These approaches are applied to identify crucial data comprised by the data mapping tables in the following section and implementation of compliance checking scenario regarding the underlying use cases.

### 5.3.3 Data Mapping Tables

The overall content of this section 5.3 represents a detailed approach of sub-processes *Determining all Data Exchange Interfaces*, *Formalizing Exchange Requirements* and *Information Mapping onto IFC Data Format* in 2.3. Now the latter mentioned sub-process can be executed by mapping the identified information provided by standards to its relating building components in IFC. Requirements for this procedure are the exact knowledge about the relating IFC entities and their distinct adherence in the mapping process. In addition, often huge data sets must be passed to enable checking for compliance with the content of the fifth section of the MBO. Therefore, a clear structure of additional data relating components and data types must be delivered to ensure a flawless workflow.

In this thesis, all additional data are mapped via property sets (see 2.2.3). bSI provides standard property sets for passing commonly used information for certain classes. These property sets start with a *Pset\_* prefix, followed by *BuildingElementCommon* and can be looked up for every particular IFC entity on “buildingSMARTf”, 2021. An exemplary object like *IfcDoor* holds specific object attributes and its relating data type, which are always defined in the class’ entity itself (see 5.12). Every door has a certain height and width independently of its use. Additional common used data can be added via *Pset\_DoorCommon* property set and can hold information as depicted in Figure 5.13. For an efficient mapping process, it is mandatory to ensure the singularity of data and prevent mapping the same data in several ways, e.g., via user-defined property sets.

#	Attribute	Type
9	OverallHeight	IfcPositiveLengthMeasure
10	OverallWidth	IfcPositiveLengthMeasure
11	PredefinedType	IfcDoorTypeEnum
12	OperationType	IfcDoorTypeOperationEnum
13	UserDefinedOperationType	IfcLabel

Figure 5.12: Standard Door Properties Included in Entity Definition (“buildingSMARTm”, 2021)

Template	PropertyName	Value
Single Value	Reference	IfcIdentifier
Single Value	FireRating	IfcLabel
Single Value	AccousticRating	IfcLabel
Single Value	SecurityRating	IfcLabel
Single Value	DurabilityRating	IfcLabel
Single Value	HygrothermalRating	IfcLabel
Single Value	IsExternal	IfcBoolean

Figure 5.13: Extraction of Additional Door Information of Pset\_DoorCommon (“buildingSMARTm”, 2021)

The strict use of common property sets for already implemented data prevents the provision of multiple equivalent mapped data. Due to the fact, property sets can only be applied to its relating (common property set) or assigned IFC entity (user-defined property set), an exact definition of specific building elements (e.g., escalator) concerning their allocation to specific IFC entities is essential (5.2.4). All project participants should agree on these export definitions in an early project stage to prevent vagueness and time delay. In addition, user-defined property sets hold information that is not comprised by common property sets. The user-defined property sets for this thesis have a *MBO\_* prefix, followed by the *BuildingElement* and ends with *Properties*, e.g. *MBO\_DoorProperties*.

Applying the techniques of the aforementioned sections, all required additional information can be detected and translated into attributes and its underlying data types to enable an assignment to IFC entities. bSI provides all information about every building element that is included in the current IFC4 version (see 5.10). Every incorporated information, common and additional, should be gathered in a table where its assignment to certain entities, underlying data types, and nomenclature can be reviewed. IFC data



The first row of Figure 5.14 represents the mapped data and its relating data types and outlines all required IFC entities for a full test of the fifth section of MBO. Even though MBO does not claim any requirements for columns or beams, these building elements are typically part of bearing structures. Due to the exported IFC entity and the filtering of components in compliance checking processes regarding the entity type, neglecting these bearing structure elements can lead to missing compliance. The blue rows represent the user-defined property sets, yellow the quantity take-off properties, and green the used common property sets. The checkmark specifies the corresponding property and IFC entity (row), respectively, and in relating columns, the property sets or quantity take of sets.

QTO sets can be looked up on “buildingSMART”, 2021 as well as common property sets. MBO states demands regarding these building elements and their dimensions. The utilization of QTO sets reduces computation time by accessing area information of windows and openings.

The whole property set table incorporates 42 additional attributes. Common property sets cover ten attributes, and 30 are mapped via user-defined property sets. Additionally, two are transferred via QTO sets. Furthermore, the number of information is reduced by five attributes concerning user-defined property sets by applying a naming convention to spaces. An extended version of the property set is available in the Appendix A with detailed information about all required information of MBO like interface and user incorporation. Additionally, mapping tables like 5.14 require a short description of all mapped properties. Additional tags can complement them to identify relating content in the standards or regulations quickly. The table’s extraction above misses these due to reasons of missing space but is included in the full document.

In BIM projects, the enrichment of additional data of building models must be well documented and communicated, and all stakeholders require access to the underlying spreadsheets gathering and explaining information in detail. These spreadsheets are characterized by a growing complexity depending on the particular project and underlying regulations. The representation of contained information in rows and columns involves the risk of errors during the mapping process and requires strict review and check of correct data assignment. In addition, the naming of attributes must be agreed upon and used for the particular case to ensure conformity of mapped data and deposited information in the hard-coded compliance test. The attributes depicted in 5.14 represent an additional exchange requirement regarding the use case of code compliance checking of the fifth section of MBO. Therefore, the compliance checking procedure in the case study must validate the existence and consistency of all required data.

### 5.3.4 Code Compliance Checking Approaches

ACCC requires reliable information to ensure that specific contents are testable and concurrent achievement of a high level of repeatability. Some operators lack a deeper understanding of the transition from standard 3D models to a semantic building model and its effects on the modeling process. Therefore, overall knowledge about proper semantic modeling is often not present and can result in models with modest quality. Additionally, for particular cases, the same semantics can be delivered in various ways inside of IFC due to a multitude of classes. For a proper code compliance workflow, key semantics must be identified on which all tests can relate. These key semantics build the basis of mandatory information for the appropriate performance of the underlying compliance tests and build the first level for all code compliance workflows. Key semantics can vary for different application fields and have to be individually identified. Therefore, the development of tests to evaluate the overall data quality of the provided building model is required to the applicability of advanced code compliance checks. In cases of additional mapped data, the correctness of mapped data in terms of assignment to all relating components, correct terminology, etc., have to be evaluated.

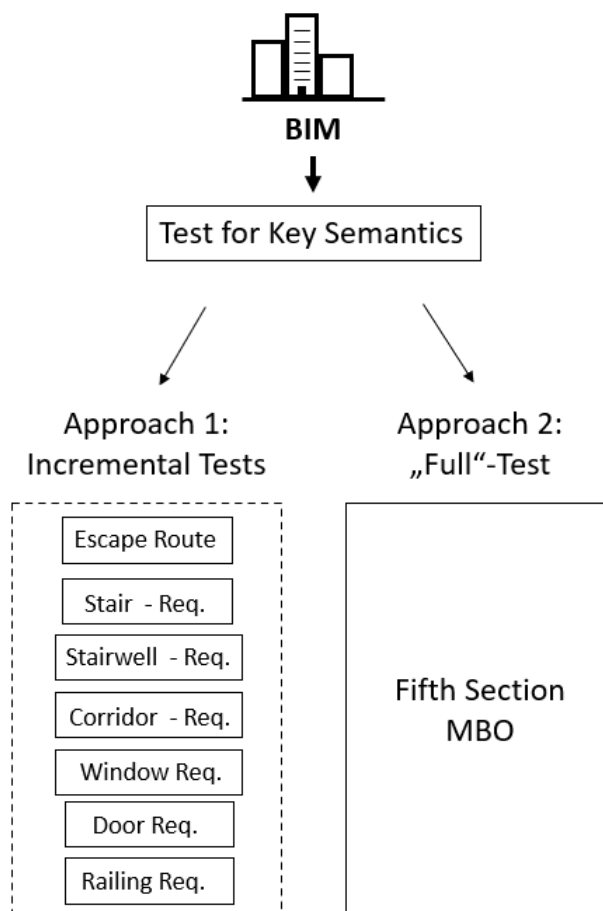


Figure 5.15: Incremental vs "Full"-Test Approach Testing for Compliance with 5th Section of MBO. Incremental Requirements are Divided into Further Sub-Tests

The followed code compliance checking level concerns the check of model quality, like clash detection tests, and planning quality of architects and engineers regarding norms and standards. Users and developers can decide whether the checking process is supposed to be incremental or that "full-" tests are preferred. "Full-" tests describe checking methods of extensive contents via a single test. This approach results in fast compliance processes but is characterized by low user engagement and a strong black box character. In addition, these tests require a lot of data due to their large and complex functionalities and often result in an inflexible manner. Therefore, it can often only be applied in late project phases. On the other hand, incremental tests represent the approach of high user engagement by utilizing a library of tests, each checking for small sub-contents of a specific use case. This can reduce the black box character to an acceptable minimum, and the user can fulfill its responsibility regarding the correctness of a model's quality.

An incremental checking process enables the use of calculated information of previous tests for subsequent tests. This possibility can reduce the amount of additional data. For example, in [MBO](#), certain building objects, like stairs, or corridors, are considered to be *necessary* if it is part of an escape route. Therefore, a test for evaluating escape routes could be applied, and all affected building elements that require the *necessary* attribute can be passed to all following tests that need this additional information, e.g., requirements concerning the composition of necessary stairwells. However, this procedure requires a library of tests that can only be built over time, and additionally, the development environment must be able to pass information from test to test. The advantages concerning incremental test scenarios are utilized in this work to facilitate a compliance checking scenario with high user involvement.

## 5.4 Technical Implementation

The overall code compliance checking procedure and its implementation depends on the development environment's capability. Therefore, the environment must provide sufficient functionalities to query and process all delivered information from the building model for conclusive issue management. An environment's functionalities depend on its underlying library of methods accessible via [API](#). A method provides a particular functionality, takes one or several input arguments, processes them, and generates specific output data. For all coding languages, the passed arguments must match the required input data type. Thus, a wide range of compatibility between methods of a library is a major issue and enhances its effectiveness.

A fundamental part of every code compliance checking test represents the filtering of building elements. All compliance tests relate to specific building components which have to be identified and extracted from the model. Enabling the user to determine all affected building elements for a particular test increases the user involvement and a test's reusability for different scenarios. Filtering properties can be conducted via certain building element types, like *IfcStair*, and further specified attributes like names, property sets, or even particular values. Varying nomenclature of building elements in different projects can be easily applied, and thus, flawless filtering of objects is assured. Applying filters results in a list of selected building components that can be used and processed in a code compliance checking process. The utilized software Solibri Office provides highly adjustable component filters. Filtering components can be conducted via extensive selection of specifiable properties.

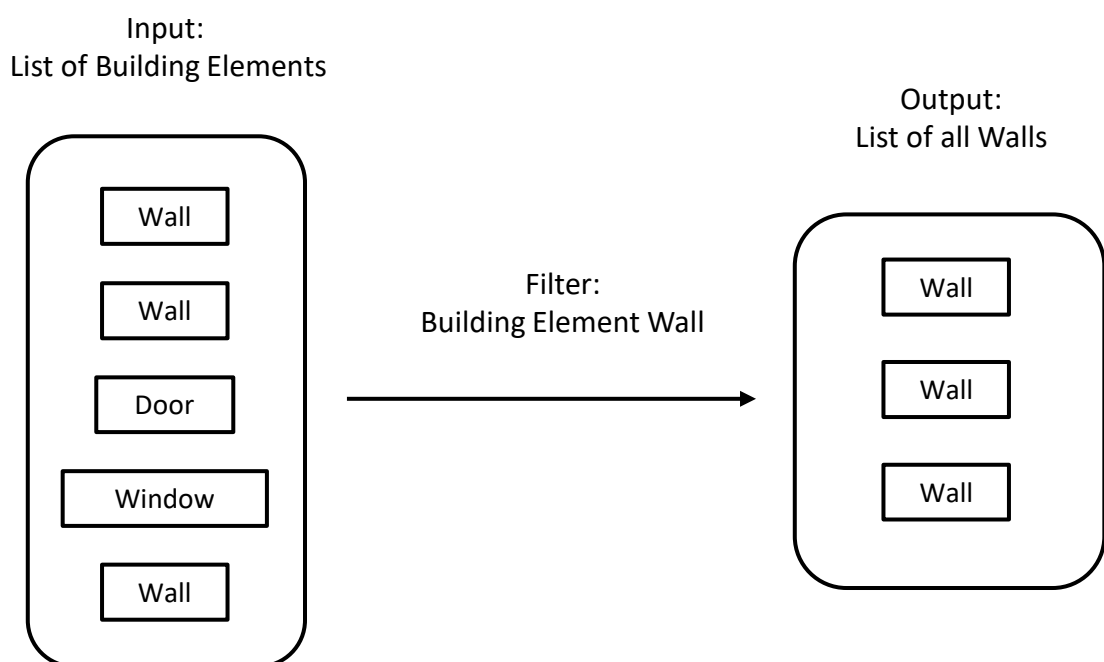


Figure 5.16: Filtering for Component Type Wall to Receive List of Wall Components for Further Processing and Calculation

For hard-coded tests, the underlying filter function represents the back-end part, and all its delivered information can be entered by the user, e.g., via Graphical User Interface (GUI) (front-end). Handling data structures like lists in hard-coded languages like Java is a common task and enables efficient methods included in standard libraries. These methods comprise arithmetical and relational operators, loops, and branching and can be utilized for filter functionalities or other processing steps.

All filtered building elements can now be investigated, and relevant information can be queried, which represents the lowest complexity level (see section 4.3.2). Significant information hold by the underlying entity can be directly accessed, or by querying respective classes, additional semantics or geometrical data can be accessed (see section 5.2). This process can be sped up by implementing its own data model and convicting all required information of data formats like IFC. An optimized data model for particular use cases like

ACCC increases the overall efficiency by decreasing complexity and querying time due to a less complicated data structure (PREIDEL, 2020). As depicted in Figure 5.17, every wall entity's properties of the filtered list of walls are examined. In this example, three different attributes are queried and can be compared to underlying claims of norms or user requirements. The query checks whether a wall is an internal or external wall, if the wall has a flammable cladding, and the value of the overall cladding thickness. Solibri Office utilizes its data format and translates required data from IFC. Therefore, the accessible information does not simply rely on the contents in IFC but also on the utilized software database. For example, via the API, all building elements are stored as Components and use an extensive library of methods to access further information. On the other hand, buildings, building stories, etc., represent their class in Solibri's data environment and provide only a few methods to query additional information.

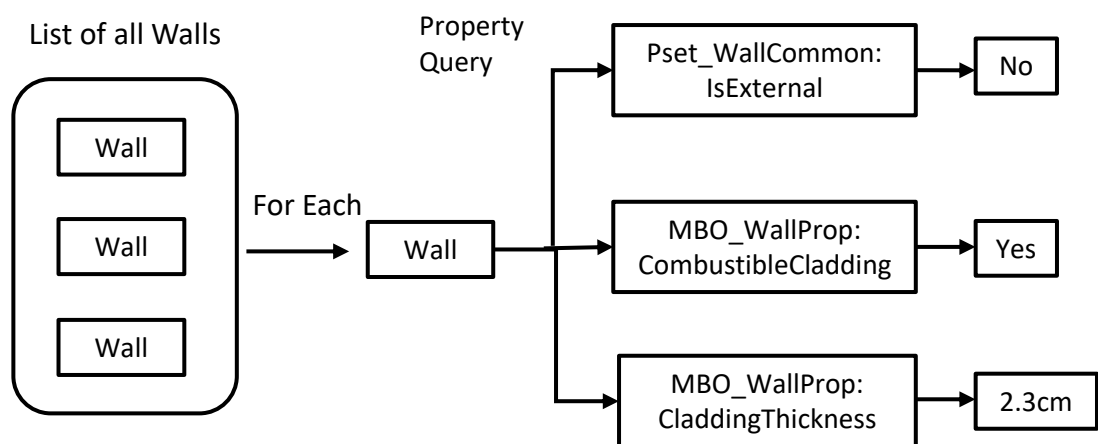


Figure 5.17: Wall's Property Query regarding its Cladding Thickness, Combustibility and External Positioning

Besides properties like names or materials, geometrical and spatial relations play a vital role in compliance checking in the AEC sector. A development environment that provides numerous functionalities for handling spatial relations facilitates broad applicability. BORRMANN, 2007 states three fundamental categories of spatial operators: metrical, directional, and topological operators and form the derivative information of 4.3.2. The former is used to determine distances between two objects. Dependent on the underlying use case, three-dimensional, plane horizontal, or vertical distances are required and represent standard spatial methods to facilitate compliance checking tests. Directional operators are utilized to determine the relative positioning of objects to each other, e.g., underneath and above or north and south. Topological operators describe relations between objects and give information about whether they touch, overlap, or contain each other (see Figure 5.18 for all possibilities). Due to large model sizes and complex computations, topological calculations often use bounding boxes for component interactions. Bounding boxes, or smallest enclosing boxes, represent an enclosing box of the relating body and therefore only represent its shape in a simple way. Detailed geometrical information of particular objects can be extracted from the underlying building model (PREIDEL, 2020). Nearly all functionalities are utilizable in Solibri Office.



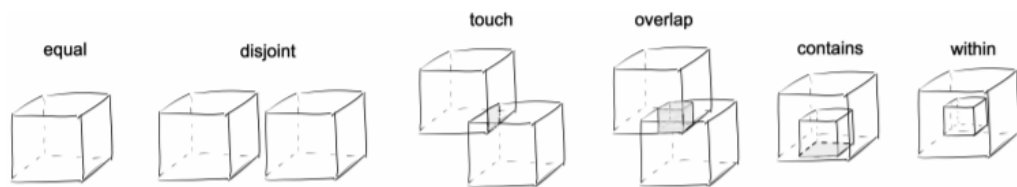


Figure 5.18: Schematic Representation of Topological Operators (PREIDEL, 2020)

These topological operators can be combined with [IFC Spatial Structure](#) and used to provide an advanced spatial relation. By assigning, e.g., *IfcSpaces* to all building's areas, building elements can be referenced with their relating spaces. Thus, all spaces link its included and bordering building elements and vice versa. By distinguishing between *containing* and *including*, and *touching* and *overlapping* topological operators, an additional semantic is generated whether a building component is part of a space or its border. Additionally, establishing a method to find the nearest building component or even a specific type is an important tool for code compliance checking and reduces a large amount of implementation work.

In addition, preprocessed data and drafted relations can be further utilized as input data for methods, e.g., escape route calculations (extended data structure see [4.3.2](#)). Methods of this large and complex scope can increase the black-box character of particular tests to a large amount and should be avoided or represent a single test instance that hands its results to subsequent tests. All results of compliance checks should be visualized in a model viewer to enhance user interaction and clarity. Introducing severity in the representation of the results of particular building elements supports the user in its decision-making and reduces the risk of false error evaluation. Solibri' [API](#) enables the user to validate intersections between building components, and therefore, a part of topological operators are utilizable. The visualization of problems and displaying different levels of severity of checking results represents a key performance of Solibri Office.

## 5.5 Summary

This chapter introduced a general approach to translate the content of standards or legal texts into computer-interpretable information to be able to use them in a code compliance checking process. For this purpose, the RASE Mark Up Technique was manually utilized. It allows the user to identify the overall structure, and thus relevant content is easier to perceive. So-called mark-up tables are used to summarize standard content clearly and enable supplementation of additional information. A basic understanding of the underlying IFC data format is of great relevance for an efficient mapping process.

For this reason, essential structures of the open data format [IFC](#) are disclosed, and thus information already contained is outlined. Detailed knowledge about the existence of content in a model is of great importance to avoid unnecessary mapping of data. In order to facilitate the extraction and subsequent mapping of information, the concept of granularity is introduced, which gives the user a starting point for how detailed a demand

or factual situation is described. Different ways of dealing with insufficient information in a code compliance checking process are revealed and discussed based on examples from the [MBO](#). The mapping of the additional information is solved via mapping tables, which contain all important information regarding the additional data and its terminology, data types, and relating building elements. Finally, the requirements for a development environment are defined, and the basic features of technical implementation of [ACCC](#) are disclosed. This includes filtering components and querying their attributes, differences of relation types, and their processing into more complex calculation methods.

The methods for identifying and processing relevant information result in this work's exchange requirements comprising the mapping table, building element allocation table, component's aggregation, and correct modeling spatial aggregation are utilized and validated in the case study. In addition, the underlying code compliance checking process uses an incremental test scenario to enhance its user involvement and validates the test building model's design quality via hard-coded tests.

## Chapter 6

# Proof of Concept

This chapter covers the outlined concept of chapter 5 and validates it on the basis of selected use cases regarding its reliability and applicability. The case study is compounded by 6 sections. The first section 6.1 reviews the additional attributes of the mapping tables. In the following section 6.2, the implemented test scenario regarding the exchange requirements and its resulting data quality is outlined. Section 6.3 covers the contents of underlying use cases §33 and §34 and comprises the general approach to test for its requirements. On this basis, the implemented checking scenarios of data quality and model quality are validated and reviewed in 6.4. Section 6.5 evaluates the achieved strengths and limitations of the overall implementation process. In the end, section 6.6 gives a summary of the before outlined content.

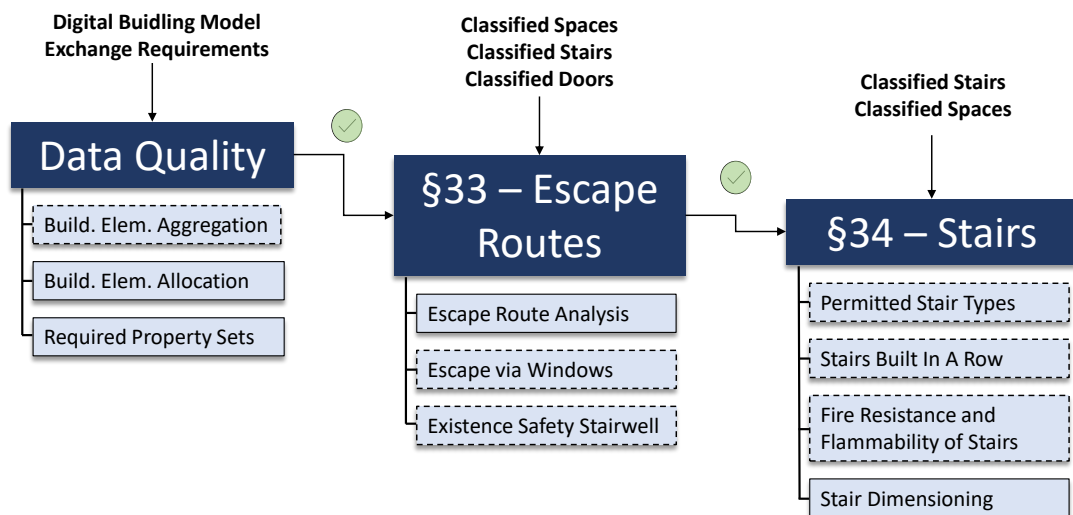


Figure 6.1: Underlying Tests Scenarios. Dotted Blocks Represent Custom Implemented Tests. Continuous Blocks Delivered from Solibri Office

### 6.1 Additional Data Identification

In section 5.3.3, 42 required information are identified to test for compliance regarding the fifth section of the MBO. Common property sets cover ten entities, and two are assigned to quantity take-off classes. The remaining 30 can further be reduced by five attributes by applying a naming convention. These reduced properties comprise information concerning space use classification, to be able to distinguish escape route parts, like corridors or stairwells, utilization units, like apartments or offices, its sub-spaces, e.g., common rooms like bedrooms or non-common rooms like bathrooms or kitchens and even different shaft types (§37, §38). This classification of spaces plays a vital role in escape route analysis

and requires strict differentiation. In particular projects, naming conventions are not always applicable, and therefore, all information must be mapped into the underlying model. All remaining properties are covered by user-defined property sets and are characterized by a *MBO\_* prefix, followed by the relating *IFC Entity* and *Properties*, e.g. *MBO\_RoofProperties*. The following outlines the identification process of additional data of *MBO* by its relating paragraphs. Therefore, the well structured text of *MBO\_RASE\_MarkUp.pdf* via RASE tags is utilized, in combination with the discussed *IFC* classes in 5.2, the principle of a requirement's granularity, and additional methods to handle insufficient detailed texts (see section 5.3). The following results are contained in *Mapping\_Table.xlsx* which links tags to *MBO\_DataTags.pdf*. Both documents are digitally attached to the Appendix A.

The first paragraph contains three additional attributes and two requirements for an interface input to cover all outlined information. Utilization units are not represented as *IfcSpaces* because it interferes with the utilization of the escape route analysis test from Solibri Office. The requirement concerning escape routes is conducted for sub-spaces and differentiated in common and non-common rooms. This attribute is not implicitly identifiable and must be mapped via an additional boolean attribute. This applies analogously to stairwells concerning their safety status. An attribute verifying whether it is a safety stairwell requires additional information to validate. Both attributes are passed via user-defined property sets. The assignment of the necessary attribute to building elements of an escape route can not be automatically passed from the escape route analysis in Solibri. Regarding stairs, all stairs are verified as necessary if it holds the *FireExit* attribute of the common property set. Therefore, the semantic is indirectly mapped to its entity, additional data is reduced, and mentioned in the relating test documentations. Furthermore, a boolean interface input is utilized to access information concerning the third section of §33. A checking-box passes the information if the fire brigade has the required rescue equipment due to a missing class in *IFC*.

§34 comprises overall 11 additional attributes and utilizes three interface inputs. Besides the covered attributes above, the attribute *RequiredSlope* is utilized to access a ramp's slope. Additionally, different stair types and components like ladders and escalators must be distinguishable. Therefore, the underlying building element types are assigned via user-defined property sets with the *Escalator*, *Ladder*, and *Retractable* property. The escalator attribute could be passed as *IfcTransportElementTypeEnum* in the relating *IfcTransportElement* entity, but in this thesis, the property set is preferred. Furthermore, properties of common property sets are required to identify the bearing structure of stairs, its flammability and fire resistance, and distinguish external and internal stairs. User input via the interface is utilized to cover the underlying building class, the stair's capacity, and the sufficient length of the stair landing.

The third paragraph contains many of the previously defined data. Overall, §35 comprises 22 additional attributes and one interface input to cover the building class. The requirement of leading to open air is represented by the *IsExternal* property of stairs. The first section requires the size of spaces which is solved by the utilization of QTO sets. *MBO* requires gross areas, thus the area of surrounding walls has to be taken into

consideration as well. The third section contains requirements concerning doors covered by common properties *SmokeStop* and *SelfClosing*. bSI states that doors assigned with a smoke stop attribute must be self-closing, so for this case, the assignment of single *SmokeStop* attribute is sufficient. Nevertheless, MBO distinguishes *AirTight* and *SmokeStop* doors, so the *SelfClosing* attribute is still required. Additionally, different wall types are claimed for specific building classes and are not implicitly identifiable. *Firewall* and *MechanicalExposure* cover these requirements concerning specific walls. Due to the required high level of detail and an increase of reproducibility, the attribute *ReachingRoof* is assigned to stairwell walls to identify if they reach the roof's membrane. Furthermore, this thesis aims to provide compliance tests capable of validating building models in early project phases. Therefore, the required flammability properties of walls, columns, and beams coverings are mapped to the relating entities instead of querying the single covering elements. The presence of stairwells safety lighting can only be identified by assignment of an additional *SafetyLightning* attribute. The smoke removal of a building is realized via openings and windows. Therefore, all components that are designed as smoke removals must hold an additional *SmokeRemoval* property to be able to identify all eligible entities. Additionally, its size (QTO property) and whether it is openable *Openable* can not be identified by the compliance checking algorithm itself and requires additional attributes.

§36 comprises overall 16 additional attributes and two interface inputs. Besides some of the previously outlined attributes, information regarding the number of a stair's steps is required if it is located inside a necessary corridor. The common property of *NumberOfThread* and *NumberOfRiser* holds detailed information regarding this claim. Additionally, necessary corridors must be separated by not lockable doors. The boolean attribute *Lockable* specifies if doors can be locked, and *SingleEscapeDirection* properties are assigned to corridor spaces due to requirements concerning its length of corridors with a single escape direction.

The first section of §37 is not evaluable by mapping additional attributes. Even if windows are categorized with an extra property to be cleanable, a relation to other building components that facilitate a cleaning process is not possible. This case relies on a user investigation by handing window entities to the user. The second section can be represented by an additional attribute assigned to windows but only relies on the mapped information, and the underlying semantic is not further processed. Therefore, this thesis does not cover the requirement due to low benefits. The last section includes requirements concerning the area of windows that are utilized as an escape route. This requires information regarding the clear inner width and height and is passed via a user-defined property set.

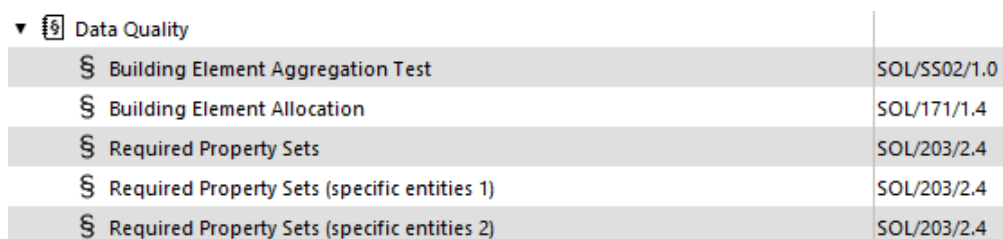
The last paragraph comprises five additional attributes. The exchange requirements regarding building element allocation and its respective entity type determine to allocate horizontal glass coverings to *IfcRoof*. Therefore, the *Walkable* attribute is only assigned to *IfcRoof*, and in addition, information concerning roofs is required if people use it for stay (*Residence*). Additionally, horizontal openings require attributes if they are covered (*SafelyCovered*), and the covering entity must be locked (*Locked*) to prevent removal.

## 6.2 Model Data Quality

Chapter 5 included the underlying exchange requirements for the checking procedure regarding the fifth section of MBO. The determined exchange requirements claim a well-implemented digital building model containing the fundamental spatial relations of a building, relating building stories and building components. Furthermore, the exported aggregated building elements must be decomposed by sub-components, and all building elements must be allocated to their determined IFC entity class. Additionally, all identified additional attributes must be present and correctly assigned. After a building model is generated and all information is enriched, the overall data quality plays a vital role for a good BIM workflow. Therefore, the abidance of the exchange requirements has to be validated to guarantee a model's data quality. As mentioned in section 3.2, a model's data quality represents the first level of the overall model quality. The implementation and use of mvdXML files facilitate validation of models concerning its provided data quality for particular MVDs. However, mvdXML requires extensive implementation effort and, as stated by POPGAVRILOVA, 2020, lacks essential functionalities resulting in insufficient results. Solibri Office provides checks to analyze underlying data quality and an open API to extend existing tests and, therefore, is utilized in this work.

Checking the compliance with exchange requirements comprises:

- Proper model structure: sufficient spatial relations and consistent component assignment - manually reviewed
- Decomposition of stairs, roofs, curtain walls and ramps - checked via SOL/SS02/1.0
- Correct building element allocation (5.1) - checked via SOL/171/1.4
- Properties existence: verify if building element holds essential property for the underlying use case - check via SOL/203/2.4
- Information location: check if property exists in required property set and if assigned to correct entity - check via SOL/203/2.4
- Property value type, syntax and overall existence: examine properties data type, comply to predefined syntax and check if values were assigned - check via SOL/203/2.4



Data Quality	
§ Building Element Aggregation Test	SOL/SS02/1.0
§ Building Element Allocation	SOL/171/1.4
§ Required Property Sets	SOL/203/2.4
§ Required Property Sets (specific entities 1)	SOL/203/2.4
§ Required Property Sets (specific entities 2)	SOL/203/2.4

Figure 6.2: Underlying Tests of Data Quality Compliance Scenario

The adherence of exchange requirements is validated with three compliance tests. The allocation of building elements and overall quality of additional attributes are validated via tests delivered by Solibri. The building element aggregation validation is conducted by implementing a test via the Solibri [API](#).

The spatial aggregation can be manually reviewed in the model of Solibri Office. The Model Tree facilitates visualizing only selected stories where the correct assignment of components to its relating story can be viewed. Automation of this process via the [API](#) can not be achieved because the building and its relating stories can not be individually queried in Solibri. Stories do not represent a physical component in Solibri and, therefore, do not comprise dimensional values. Building elements are represented by the *Component* class inside of Solibri and supply numerous methods to query additional data. Stories rely on the *BuildingStorey* class and do only provide information about a stories' name and its relating building components. A building's story can only be accessed via single components to get information about its location.

The aggregation of components can be manually examined or tested via custom implemented test *Building Element Aggregation Test - SOL/SS02/1.0* in the [API](#) which can be found in the Attachment [A](#). This functionality can be implemented via the [API](#), but Solibri handles sub-components of aggregations like main components. For example, a roof has its own *IfcRoof* entity and is aggregated of 4 *IfcPlates* as sub-components. These sub-components are recognized as individual roof entities and pass the filter if roofs are selected as components. Sub-components do not have further aggregation units and, therefore, are marked as fault components. The occurrence of errors can be easily prevented by adjusting the filter and excluding, e.g. *IfcEntitys* of type *IfcSlab* (see Figure 6.3).

State	Component	Property	Operator	Value
Include	Stair	IFC Entity	None Of	[IfcStairFlight]
Include	Roof	IFC Entity	None Of	[IfcSlab]

Figure 6.3: Adjustment of Filter in Element Aggregation Test to Generate Proper Results

Another important issue is the correct building element allocation which can be automatically checked with Solibri test *Component Property Values Must Be Consistent - SOL/171/1.4* which was renamed to *Building Element Allocation*. This test enables the user to check if all components are, e.g., of the same *IfcType* and incorrectly assigned components can be easily identified.

*Required Property Sets - SOL/203/2.4* validates if the digital building model contains specific property sets and the correctness of value assignment. As an input parameter, all components that are supposed to be checked have to be assigned to filters in *Checked Components*, and in *Property Sets*, all requirements concerning particular property sets and properties can be adjusted. For example, all stairs require (*Must exist*) the *Pset\_StairCommon* property set which must hold the *FireExit* property and must be assigned as *True* (see 6.4). To ease and accelerate this process, platforms like BIMQ provide excel files to allow an automated rule set up.

The screenshot shows the 'PARAMETERS' window with the following data:

State	Component	Property	Operator	Value
Include	Stair	IFC Entity	Matches	IfcStair
Include	Space			
Include	Door			
Include	Window			
Include	Wall			
Include	Roof			

Component	Property Set	Property	Value Exists	Value Conditions	Visualization
Stair	Pset_StairCom...	FireExit	Must exist	X = True	
Door	Pset_DoorCom...	FireExit	Must exist	X = True or False	
Window	Pset_Window...	FireExit	Must exist	X = True or False	
Window	Pset_Window...	IsExternal	Must exist	X = True or False	

Figure 6.4: Parameter Set Up for *Required Property Sets - SOL/203/2.4*

The parameter set up in Solibri Office is flexible and enables the user to configure all requirements according to his claims. *Checked Components* of Figure 6.4 represents the filters for the following property set check. These filters, e.g., can be adjusted by filtering only components with particular names, classification, or all other properties. Due to this detailed adjustment, the test scenario can be highly detailed, and a single component type can be tested for different requirements. Like for all other testing scenarios, Solibri Office visualizes all errors with the underlying components.



## 6.3 Use Cases

This section outlines the content and requirements of this work's underlying use cases to evaluate the approach's capacity. The first use case covers §33 and its requirements concerning the first and second escape way. This paragraph is chosen as a use case due to its complexity in terms of escape route calculations and its importance in the following paragraphs. The second use case is § 34, comprising all requirements of necessary stairs. Necessary Stairs are part of an escape route and, therefore, similar to the other paragraphs representing further requirements for escape route elements like corridors or stairwells. After the use cases' contents and requirements are outlined, its implementation in Solibri Office is demonstrated, and precise test documentation is provided in the Appendix A.

### 6.3.1 §33 - First and Second Escape Way

§33 is the first paragraph of the fifth section of the MBO and claims requirements for the first and second escape ways. It comprises three sub-sections and represents a central object in the following paragraphs of §34-§37. §34 to §37 are parts of escape routes and state additional, more detailed requirements concerning its properties and design. The following depicts the original quote of §33 of the MBO and a short explanation of essential information:

(1) Für Nutzungseinheiten mit mindestens einem Aufenthaltsraum wie Wohnungen, Praxen, selbstständige Betriebsstätten müssen in jedem Geschoss mindestens zwei voneinander unabhängige Rettungswege ins Freie vorhanden sein; beide Rettungswege dürfen jedoch innerhalb des Geschosses über denselben notwendigen Flur führen.

The first part introduces the term of utilization units and only specifies it by a list of examples, like apartments or practices. Utilization units with at least one common room require two independent escape routes leading to open air in every story. These escape routes can use the same corridor (horizontal escape way), but requirements regarding vertical escape routes and their independence are further specified in the second part of §33.

This first part can be tested by filtering *IfcSpaces* and querying the property of *CommonRoom* to verify if it requires two or one escape route. The complex escape route calculation requires the before identified spaces and, additionally, the destination components leading to open air, like a door with, e.g., an *IsExternal* property. The escape route must pass doors connecting two separated spaces and must not omit walls. Therefore, all rooms in a building must be represented by *IfcSpaces* to guarantee a valid escape route space. A graph can then be generated, and components like stairwells or stairs can be classified or marked as necessary components.

(2) Für Nutzungseinheiten nach Absatz 1, die nicht zu ebener Erde liegen, muss der erste Rettungsweg über eine notwendige Treppe führen. Der zweite Rettungsweg kann eine weitere notwendige Treppe oder eine mit Rettungsgeräten der Feuerwehr erreichbare Stelle der Nutzungseinheit sein. Ein zweiter Rettungsweg ist nicht erforderlich, wenn die Rettung über einen sicher erreichbaren Treppenraum möglich ist, in den Feuer und Rauch nicht eindringen können (Sicherheitstreppenraum).

The second part refers to the previous one and specifies more detailed requirements for vertical escape routes relating to the first and second escape routes. Primary escape routes must use a stair which are therefore classified as necessary stairs. All essential components of an escape route are categorized as necessary. The second escape route can be a different stair (separated vertical escape routes, but accessed via the same corridor) or a location in a utilization unit that the fire brigade can reach via rescue equipment. Additionally, the existence of a second escape route is not claimed if the building has a safety stairwell. This term of safety stairwells or further requirements are not specified in the [MBO](#).

Invalid input components of the escape route check, which do not have a required second escape route, can then be tested for the existence of windows enabling a secondary escape route or the presence of a safety stairwell in the building. Valid escape windows can be identified by holding information about whether they are external, and it is supposed to be a *FireExit*. These windows can be queried by being part or close to the relating *IfcSpace*. Safety stairwells can be identified via the naming or implementation of an additional filter adjustable by the user. An additional filter is a more flexible approach and should be favored for implementation due to resulting problems utilizing a hard-coded required nomenclature. In this work, an incremental test scenario implementation is preferred, and therefore, the requirements regarding escape windows and safety stairwells are split into two separate tests.

(3) Gebäude, deren zweiter Rettungsweg über Rettungsgeräte der Feuerwehr führt und bei denen die Oberkante der Brüstung von zum Anleitern bestimmten Fenstern oder Stellen mehr als 8 m über der Geländeoberfläche liegt, dürfen nur errichtet werden, wenn die Feuerwehr über die erforderlichen Rettungsgeräte wie Hubrettungsfahrzeuge verfügt. Bei Sonderbauten ist der zweite Rettungsweg über Rettungsgeräte der Feuerwehr nur zulässig, wenn keine Bedenken wegen der Personenrettung bestehen.

The last part specifies the requirements concerning rescue via the fire brigade's life-saving equipment. Locations or windows which are more than 8 m above the ground are only permitted as secondary escape routes if the fire brigade has the required equipment. Additionally, special buildings are mentioned, and rescue via life-saving equipment is only permitted for special cases. This work and its implemented tests do not cover special structures. The before-mentioned requirements regarding the fire brigade do not relate to

any object representable in a building model. Therefore, it is transmitted as an interface input by the user.

Filtering spaces to analyze their escape route can be achieved in several ways, e.g., their naming. This filtering is a flexible solution for varying project requirements and therefore represents a proper solution. Additionally, properties of *CommonRoom* and *SafetyStairwell* have to be assigned to spaces to verify essential information concerning the underlying component and test scenario. *IsExternal* and *FireExit* are determined semantics to identify valid escape windows. Therefore, four properties have to be mapped into the underlying building model to test for compliance with §33.

## Test Scenario Implementation

This test scenario utilizes three compliance tests to validate the underlying building model regarding §33. Therefore, the test *Escape Route Analysis - SOL/179/4.3* is used and two additional tests (Second Escape Route via Window & Second Escape Route via Safety Stairwell) are implemented via the Solibri API, covering escape routes via windows and safety stairwells (see Figure 6.5).

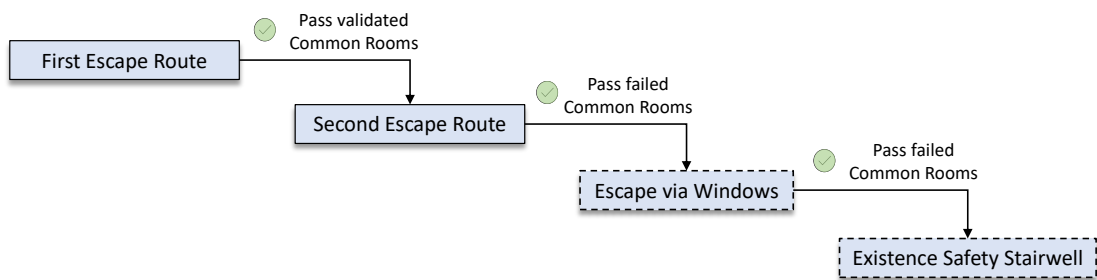


Figure 6.5: Test Scenario of Common Rooms. Continuous Blocks Represent Tests Provided by Solibri. Dotted Blocks Represent Custom Implemented Tests via API

Solibri Office provides the check *Escape Route Analysis - SOL/179/4.3* to calculate and verify a space's escape route leading to a user-defined exit door. As input, the test requires the underlying spaces by utilizing its classification functionality. All rooms can be classified and loaded into the test's filter. Additionally, all stairs and exit doors have to be entered via classification and therefore, the test only verifies devised escape routes. Classifications can be set up and saved in Solibri and loaded into desired projects independently. This facilitates reusable and fast test usage, despite a constantly changing building model during a project. The investigated room entities are then forwarded to the following test to verify whether escape windows or safety stairwells are present.

Solibri Office utilizes incremental test scenarios by providing the use of gatekeeper rules. These rules regulate conditions under which the following test receives particular components. Therefore, the *Escape Route Analysis - SOL/179/4.3* can be used as a gatekeeper rule and adjusted, determining which building elements are handed over to the following test (see 6.6). Only components of the main default filter can be passed to a gatekeeper rule's sub-rules. Thus handing over components to another rule is possible

but limited by this fact. The functionality of nesting tests is unlimited, and large nesting chains can be created to enable highly flexible and user-central compliance scenarios. Nevertheless, nesting must not result in confusing test structures and is only applicable in a controlled manner.

The screenshot shows a software interface for configuring a rule component. At the top left is an 'INFO' icon. The main area is divided into three sections: 'Name', 'Description', and 'Sub Rule Options'. The 'Name' field contains 'Escape Route Analysis'. The 'Description' section has an 'Edit' icon and a text box containing: 'This rule checks that it is possible to exit safely from the building in case of fire or other emergency. The building must have sufficient amount of suitable located exit passage ways that have sufficient capacity, so that exit time is not dangerously long.' The 'Sub Rule Options' section contains four radio button options: 'Check all model components, if passed', 'Check all model components, if issues', 'Check only failed components', and 'Check only passed components', with the last option selected.

Figure 6.6: Adjustment of Gatekeeper Rule Component Handover

The test scenario for §33 comprises three different tests in which the *Escape Route Analysis - SOL/179/4.3* is utilized several times to test for the first and second escape route. Both tests are nested and validate all space components passing the first escape route test for the existence of a second route. All space components failing this second escape route analysis have to be checked for a safety stairwell in the building or valid escape windows (see Figure 6.7). The test scenario includes incremental tests and is in chronological order extended by an additional test. This concept is chosen due to the mechanism of Solibri Office that occurring errors in between the test chain are not displayed in the results and the overall test chain disappears from the working interface. Extending all incremental scenarios step by step enables the user to keep track of occurring errors. The primary escape route analysis is set up with a high travel distance, and its destination door leads to open air. This is set up to verify the overall escape route of the building, and later on, the escape route to the stairwell can be tested with the required distance of 35m (§35 (2)) via an additional escape route test. The fundamental test *Escape Route Analysis - SOL/179/4.3* is not able to calculate an escape route over ladders or ramps. Filtering ladders or ramps in the Vertical Access was not recognized in test building models, and escape route calculations failed. Therefore, the underlying test scenario only validates stairs. A detailed individual report of a test's functionality and limitations can be found in the test documentation in the Attachment [A](#).

▼ § Common Room	
§ First Escape Route Analysis	SOL/179/4.3
▼ § Second Escape Route Analysis	SOL/179/4.3
§ Escape Route Analysis	SOL/179/4.3
▼ § Escape Route Analysis via Windows	SOL/179/4.3
▼ § Escape Route Analysis	SOL/179/4.3
§ Second Escape Via Windows	SOL/SS04/1.0
▼ § Escape Route Analysis via Safety Stairwell	SOL/179/4.3
▼ § Escape Route Analysis	SOL/179/4.3
▼ § Second Escape Via Windows	SOL/SS04/1.0
§ Second Escape Via Safety Stairwell	SOL/SS06/1.0
▼ § Escape Route Analysis via Safety Stairwell - In Range?	SOL/179/4.3
▼ § Escape Route Analysis	SOL/179/4.3
▼ § Second Escape Via Windows	SOL/SS04/1.0
▼ § Second Escape Via Safety Stairwell	SOL/SS06/1.0
§ Escape Route Analysis - Distance to Safety Stairwell	SOL/179/4.3
▼ § Non Common Room Escape Route Analysis	
§ First Escape Route Analysis	SOL/179/4.3

Figure 6.7: Test Scenario to Test for Compliance with Paragraph 33 of [MBO](#)

### 6.3.2 § 34 - Stairs

Paragraph 34 claims requirements concerning necessary stairs which are part of an escape route. It declares different prohibited stair types and several design requirements regarding underlying building class. The following outlines the original quote of §34 and a short explanation of essential information:

(1) Jedes nicht zu ebener Erde liegende Geschoss und der benutzbare Dachraum eines Gebäudes müssen über mindestens eine Treppe zugänglich sein (notwendige Treppe). Statt notwendiger Treppen sind Rampen mit flacher Neigung zulässig.

Every story requires at least one stair if it is above ground level. Instead of stairs, ramps with a slight slope are accepted as well. This requirement is already checked for a well set up of §33 via utilizing *Escape Route Analysis - SOL/179/4.3*. The escape route analysis test by Solibri can not calculate an escape route over ramps and is mentioned as a limitation of the existing tool in the test documentation.

(2) Einschiebbare Treppen und Rolltreppen sind als notwendige Treppen unzulässig. In Gebäuden der Gebäudeklassen 1 und 2 sind einschiebbare Treppen und Leitern als Zugang zu einem Dachraum ohne Aufenthaltsraum zulässig.

The second part defines invalid stair types like retractable stairs and escalators. Additionally, the building class of the underlying building determines if retractable stairs and ladders are acceptable as access to attics with a common room. This only applies to building classes 1 and 2.

The core of this test scenario still is the escape route analysis which requires all necessary stairs of the building. These classified stairs have to be split for the underlying tests of §34, so only stairs of a common stairwell are filtered. By passing stairs per stairwell, the highest stair and its connection to the attic is identifiable. Retractable stairs holding the property of *Retractable* are prohibited for particular building classes with common rooms in the attic. Additionally, the attic is represented by *IfcSpaces* and holds the *CommonRoom* property. The escape route test scenario already determines a possible escape route from the attic over the stairs. Entities like escalators and ladders can be verified by the property *Escalator* or *Ladder*. The underlying building class can be assigned to the underlying *IfcBuilding* entity but will be implemented as a user input for this work.

- (3) Notwendige Treppen sind in einem Zuge zu allen angeschlossenen Geschossen zu führen; sie müssen mit den Treppen zum Dachraum unmittelbar verbunden sein. Dies gilt nicht für Treppen
  1. in Gebäuden der Gebäudeklassen 1 bis 3,
  2. nach § 35 Abs. 1 Satz 3 Nr. 2

Necessary stairs have to be built in a row and connect to every story. Additionally, they must connect to the stair leading to the attic of the building. An exception for this are buildings of building classes 1 to 3 or two-story utilization units smaller than 200 square meters and with different escape ways on each floor.

This case is hard to generalize due to its low level of granularity. Further specifications of in-row-built stairs are not outlined and, therefore, hard to detect. Again, all necessary classified stairs are used as input stairs but have to be split, so the user enters only stairs of a common stairwell. A test of its horizontal distance and a user-adjustable threshold might be a possible solution to cover most of the occurring cases. Their connection to all stories is validated before in the escape route analysis test. Additionally, input via the interface concerning the building class is again implemented. The second exception concerning the utilization unit represents a rare case and is neglected in the implementation of this work.

- (4) Die tragenden Teile notwendiger Treppen müssen
  1. in Gebäuden der Gebäudeklasse 5 feuerhemmend und aus nichtbrennbaren Baustoffen,
  2. in Gebäuden der Gebäudeklasse 4 aus nichtbrennbaren Baustoffen,
  3. in Gebäuden der Gebäudeklasse 3 aus nichtbrennbaren Baustoffen oder feuerhemmend sein. Tragende Teile von Außentreppen nach § 35 Abs. 1 Satz 3 Nr. 3 für Gebäude der Gebäudeklassen 3 bis 5 müssen aus nichtbrennbaren Baustoffen bestehen.

Part 4 claims requirements concerning the fire resistance and flammability of materials of the bearing structure of necessary stairs. Building classes 1 and 2 have no conditions, and external and internal stairs are distinguished. Regarding exterior stairs, the bearing elements must be noncombustible materials independently of the underlying building classes 3 to 5. Internal stairs must be fire resistant and noncombustible in building class 5, in building class 4 be incombustible and in building class 3 either be noncombustible or fire resistant.

Internal and external stairs can be distinguished by accessing the *IsExternal* property. Additionally, its fire resistance is stored in the *FireRating* attribute, and its flammability is passed by the property *SurfaceSpreadOfFlame*. Both properties can be mapped to all sub-entities with their specific entity of the stair, and all aggregated elements are queried to determine the most critical component. Applying another approach by manually specifying the critical element by the user during modeling and the resulting fire rating and flammability are mapped to the single stair object. This work follows the approach of mapping the flammability and fire resistance of the most critical element of the stair to its stair object. The underlying building class is again passed by user input and distinguishes all occurring cases.

- (5) Die nutzbare Breite der Treppenläufe und Treppenabsätze notwendiger Treppen muss für den größten zu erwartenden Verkehr ausreichen.

The usable stair width must match the maximum possible traffic. The [MBO](#) does not state any requirements but can be found in DIN 18065.

- (6) Treppen müssen einen festen und griffsicheren Handlauf haben. Für Treppen sind Handläufe auf beiden Seiten und Zwischenhandläufe vorzusehen, soweit die Verkehrssicherheit dies erfordert.

A stair must have at least one fixed handrail or both sides, if the transport safety claims it. Like part 5, no further requirements are stated.

- (7) Eine Treppe darf nicht unmittelbar hinter einer Tür beginnen, die in Richtung der Treppe aufschlägt; zwischen Treppe und Tür ist ein ausreichender Treppenabsatz anzuordnen.

A stair requires a sufficient distance to a door opening in its direction. The provided *Solibri Accessible Stair Rule SOL/210/3.1* test can verify requirements of (5) to (7).

Stairs are filtered in test scenario for § 33 via classifications and are handed over to *Escape Route Analysis - SOL/179/4.3*. Therefore, these stairs are used again via their classification by the *FireExit* property to be evaluated against design requirements stated by § 34. To facilitate this process, six other properties are mapped to identify partially prohibited stair types by *Ladder* and *Escalator* attribute. These data are mapped

to *IfcTransportElements* and *IfcBuildingElementProxy* due to the missing ladder class in IFC. Additionally, stairs and ladders must hold information about its flammability (*SurfaceSpreadOfFlames*) and fire resistance (*FireRating*) and stairs must be identifiable as retractable (*Retractable*). The *CommonRoom* property must be assigned to rooms in the attic to identify common rooms in the attic.

## Test Scenario Implementation

§34 is implemented via four compliance tests. The first three of 6.8 are custom implemented tests via the Solibri API and the last test is provided by Solibri. All tests receive the classified stairs of the escape route scenario via its main default filter, and the user can enter the building class via interface.

▼ Stair	
§ Permitted Stair Type Check	SOL/SS05/1.0
§ AllStairsInOneStairwell	SOL/SS01/1.0
§ FireResistance and Flammability of Necessary Stairs	SOL/SS03/1.0
§ Accessible Stair Rule	SOL/210/3.1

Figure 6.8: Test Scenario to Test for Compliance with Paragraph 34 of MBO

## 6.4 Validation

This section validates the approach's quality regarding its mapped data and the relating checking process concerning its consistency to enable a compliance check of the outlined use cases. Additionally, the performance of implemented test scenario is assessed to check for compliance with §33 and §34 in terms of its applicability to verify a building model. In the beginning, the exchange requirements for the underlying compliance scenarios are validated, and wrong assigned components are identified to ensure a consistent building model. In the following step, the building model, which comprises errors conflicting with several requirements of the MBO, is checked for conformity with both paragraphs. All errors have to be identified, and a corrected building model will undergo the same testing procedure to validate the checking procedure's quality.

### 6.4.1 Validation of Data Quality

Section 6.2 outlined the overall checking structure to verify key semantics for §33 and §34. Occurring errors and relating components are described in the checking results and can be quickly reviewed by the user (see Figure 6.9). All issues can be communicated via BIM Collaboration Format (BCF) and adjusted by relating person in charge in the underlying modeling software.



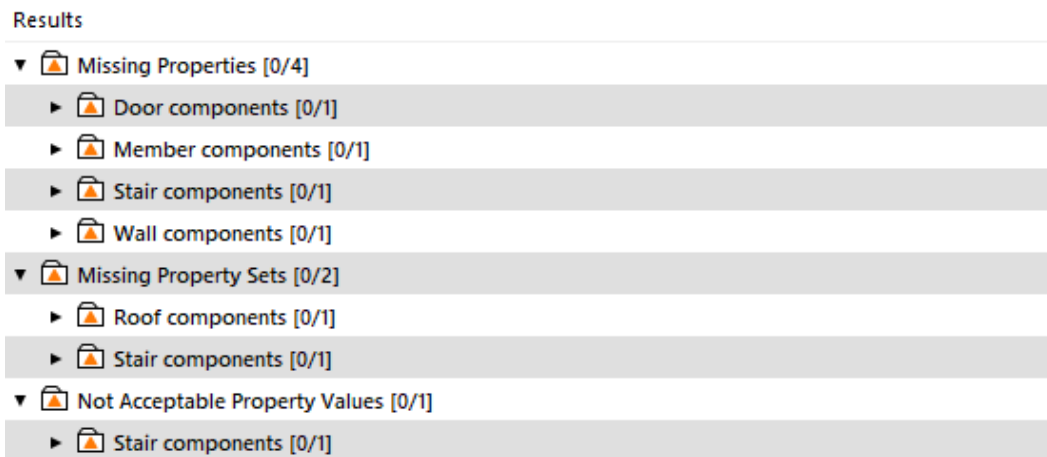


Figure 6.9: Results of Data Quality Scenario for Following Compliance Check of §33 and §34

The side of the building's entrance is not supposed to have windows used for rescue, and therefore, the property *FireExit* is *false* for all windows on that side of the building. Additionally, the stairwell is not designed as a safety stairwell, and all stairwell spaces are marked as *false* regarding the *SafetyStairwell* property. All these requirements can be adjusted and verified via the *Required Property Sets - SOL/203/2.4* test.

In general, the *Building Element Aggregation Test - SOL/SS02/1.0* enables a fast setup to check for aggregation of building elements. Additionally, a correct element allocation can be tested by utilizing *Component Property Values Must be Consistent - SOL/171/1.4*. Both tests can be adjusted to the user's requirements and help to check crucial aspects of a model's data quality. However, checking for the existence of certain properties and property sets with additional correct assignment represents a complex checking task. *Required Property Sets - SOL/203/2.4* can be flexibly adjusted via component filters. Additionally, adding several compliance tests of the same type facilitates an agile workflow to handle all occurring scenarios in this work. Many design errors regarding this MBO section can only be detected by plain information enrichment regarding particular components. This results from missing detailed requirements, e.g., regarding the design of safety stairwells, or for a computer undetectable functionalities, like whether a stair is retractable or not. Large models with huge amounts of data require extensive data management to ensure an accurate information assignment and a proper resulting code compliance checking procedure.

## 6.4.2 Validation of Design Quality

The employed building model comprises the following errors which have to be detected:

- Insufficient escape route available. Find windows for second escape route or check for existence of safety stairwell
- Retractable stairs as necessary stair
- Insufficient flammability of stairs

### Escape Route

All common rooms, e.g., living rooms, require two escape routes leading to open air. The underlying building model only has a single entrance, so the failing space components of the second route analysis are checking for rescue windows or the presence of a safety stairwell.

▼ [🔍] Common Room			
§ First Escape Route Analysis	🔑		OK
▼ § Second Escape Route Analysis			
§ Escape Route Analysis	🔑	⚠️	
▼ § Escape Route Analysis via Windows			
▼ § Escape Route Analysis			
§ Second Escape Via Windows			⚠️
▼ § Escape Route Analysis via Safety Stairwell			
▼ § Escape Route Analysis			
▼ § Second Escape Via Windows			
§ Second Escape Via Safety Stairwell			⚠️

Figure 6.10: Result Overview of Test Scenario for Common Rooms Checking for First and Second Escape Route

Figure 6.10 depicts the result overview of the test scenario for common rooms. The first escape route analysis test is passed and displaced with a green *OK* sign. The second escape route analysis test failed and is marked with the highest severity due to missing escape routes. The third test chain verifies all failed space components and detects 30 rooms that can not be evacuated via the relating windows. The test provides the possibility to choose if the fire brigade has necessary life-saving equipment to enable a rescue via windows 8m above ground. All space components with windows designated for rescue now pass the test and do not require a second escape route. The remaining 24 common rooms on the north side miss their second escape route if the building does not have a safety stairwell. The current stairwell design is not intended to be a safety stairwell. Therefore the test can not find any safety stairwell in the building (see 6.11).

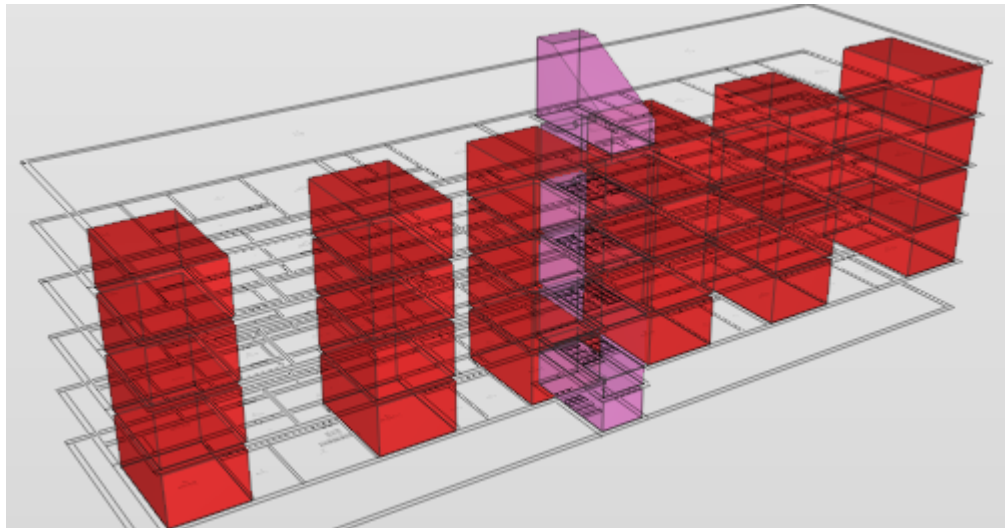


Figure 6.11: Visualization of Common Rooms missing Second Escape Route and Identified Stairwell

All issues and affected components are communicated and adjusted to guarantee a sufficient escape route for all common rooms. Rescue via windows on the north side is not possible, so that the stairwell will be designed as a safety stairwell. The same test scenario validates the adjusted version of the building model. Additionally, the length of the escape route leading to the safety stairwell is investigated and limited to 35m. Both tests regarding the safety stairwell are passed, and the building model fulfills all escape route requirements of § 33 (see Figure 6.12).

▼ [9]	Common Room				
§	First Escape Route Analysis				OK
▼	§ Second Escape Route Analysis				
§	Escape Route Analysis			△	
▼	§ Escape Route Analysis via Windows				
▼	§ Escape Route Analysis				
§	Second Escape Via Windows			△	
▼	§ Escape Route Analysis via Safety Stairwell				
▼	§ Escape Route Analysis				
▼	§ Second Escape Via Windows				
§	Second Escape Via Safety Stairwell				OK
▼	§ Escape Route Analysis via Safety Stairwell - In Range?				
▼	§ Escape Route Analysis				
▼	§ Second Escape Via Windows				
▶	§ Second Escape Via Safety Stairwell				OK
▼ [9]	Non Common Room Escape Route Analysis				
§	First Escape Route Analysis				OK

Figure 6.12: Test Validates New Assigned Safety Stairwell and No Second Escape Route Is Required

### Stair Requirements

Stairs represent an essential part of the escape route analysis and are a user input parameter for *Escape Route Analysis - SOL/179/4.3*. Therefore these stairs are further tested for requirements regarding different prohibited stair types, fire resistance, and flammability properties, and if all stairs are built in a row. Additionally, MBO claims undefined limitations for stairs regarding its dimensions.

▼ [9]	Stair				
§	Prohibited Stair Type Check			△	
§	AllStairsInOneStairwell				OK
§	FireResistance and Flammability of Necessary Stairs			△	
§	Accessible Stair Rule			△ △	

Figure 6.13: Result Overview of Test Scenario for Stairs

Figure 6.13 shows the result overview of the test scenario for stairs. The test identified two retractable stairs and an escalator which were selected by the user to test for escape route parts but are prohibited in the underlying building class 4 (see Figure 6.14). Besides stairs, the test can verify ladders and escalators if the necessary properties are assigned. Additionally, the fire resistance and flammability do not comply with the claims regarding building class 4. The last test enables to test many stair properties which cover the last

three parts of §34, but because of missing limitations of the MBO further investigations are neglected.

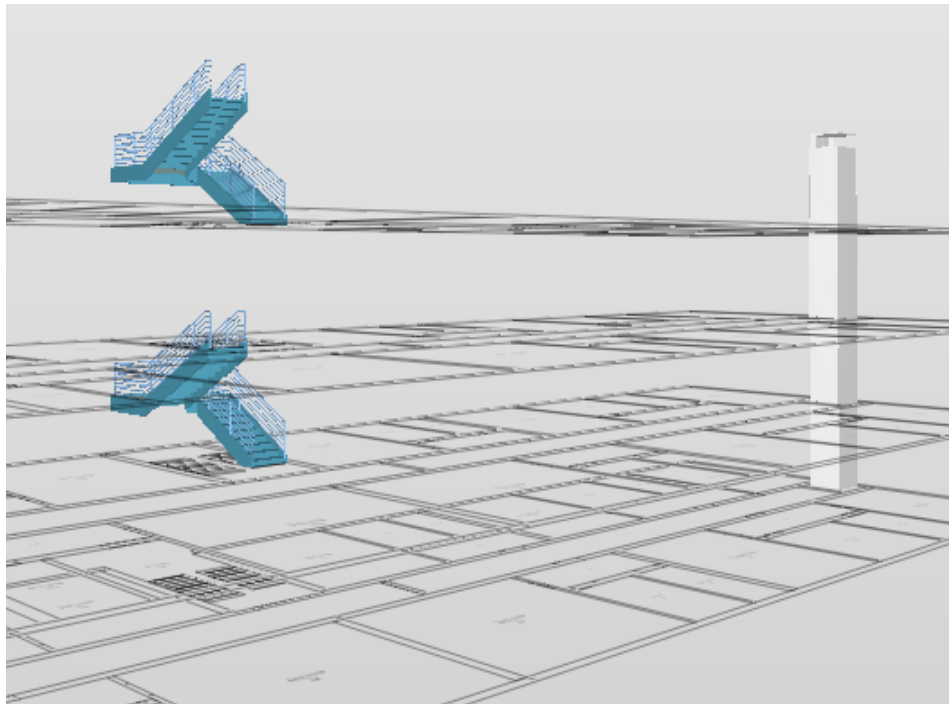


Figure 6.14: Visualization of Prohibited Stair Types and Elevator as Escape Route Parts in Building Class 4

All occurring errors are again communicated, and the stairs are not designed as retractable stairs anymore. Additionally, all stairs are planned as noncombustible to comply with the flammability claims of MBO regarding building class 4. Test results Figure 6.15 depicts that all tests are passed, and the building model fulfills all requirements.

Stair	
§ Permitted Stair Type Check	OK
§ All Stairs In One Stairwell	OK
§ FireResistance and Flammability of Necessary Stairs	OK

Figure 6.15: Test Validates New Assigned Property Values Concerning Flammability and Types of Stairs

## 6.5 Approach Evaluation and Limitations

The validation results outline the capacity of the overall compliance checking procedure and demonstrate an effective approach to test for requirements claimed by norms or other regulations like the [MBO](#). The following evaluation reviews the strengths and limitations of the development environment of Solibri Office, examines the resulting black-box character of the checking scenarios, and discusses the context of modeling software and the overall modeling process regarding its effect on the compliance checking process.

Solibri provides several tests to check for particular parts of the fifth section of [MBO](#). The [API](#) enables the implementation of user-defined rules to complement specific contents. The implemented tests of this work are explained in test documentation (see [A](#)), which individually covers every test's functionality and limitations. Test documentations give the user a better understanding of the applied compliance tests and reduce the chance of undetected errors or wrong utilization. Nevertheless, this does not apply to the provided tests by Solibri. Tests with a more straightforward functionality, like *Required Property Sets - SOL/203/2.4* are easier to understand and might be acceptable with missing documentation. On the other hand, complex tests like *Escape Route Analysis - SOL/179/4.3* require an extensive understanding of the overall Solibri functionalities, and essential modeling requirements can only be detected by trial and error. For example, [MBO](#) claims the existence of escape routes for utilization units, e.g., apartments. If the overall utilization unit is represented by an *IfcSpace* and its sub-spaces are present as well, *Escape Route Analysis - SOL/179/4.3* does not recognize the walls and calculates escape routes leading through walls (see [Figure 6.16](#)). Additionally, limitations regarding uncovered cases like ramps and ladders are not mentioned and can only be tested by the user.

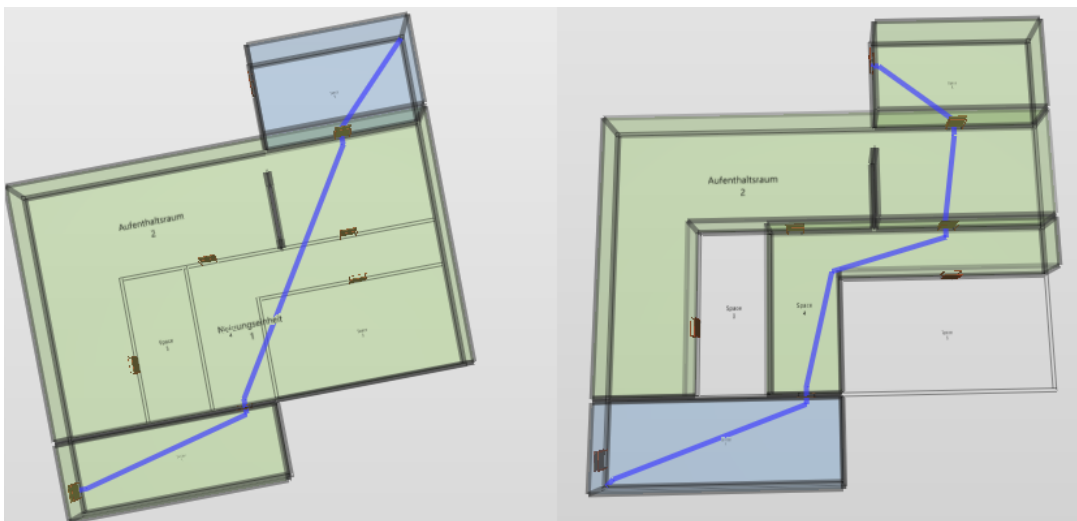


Figure 6.16: Left Example: Represents Overall Utilization Unit and Includes its Sub-Spaces e.g., Living Room. Right Example: Model Only Contains Sub-Spaces and Generates Correct Escape Route

The first compliance scenario utilizes the Gatekeeper functionality. Gatekeeper rules are a strong tool and increase the overall depth of testable content by creating test chains of several checks. However, this functionality comes along with constraints that partially interfere with the compliance checking process. The first limitation of Gatekeeper rules is that only specified components of the main default filter can be passed by condition to following rules. On the one hand, this limitation prevents confusing checking chains, but regarding many checking procedures, it would be helpful to choose which components or objects are passed. For example, handing over the calculated escape route graph and classifying affected building components would automate the overall process. Nevertheless, Solibri Office created a proper user involvement by utilizing its flexible filter functionalities. This enables an agile workflow, and the person in charge is always aware of which components are entering the checking process. Additionally, chaining compliance tests via Gatekeeper rules facilitate a highly incremental checking process. Requirements claimed by §33 and §34 are split up into different tests, resulting in smaller and simpler tests. Regarding the underlying black-box tests, this again reduces the risk of undetected errors or wrong application. The most significant limitation of Gatekeeper rules is the disappearing of rule sets if a checking chain is failing in between. This is only symbolized by a black bar in the error severity, but the problem is not identifiable (see Figure 6.17). A new setup of the overall checking chain is only possible in the *Ruleset Manager*. This problem could not be solved and is just counteracted by extending the test chains step by step to determine issues early.

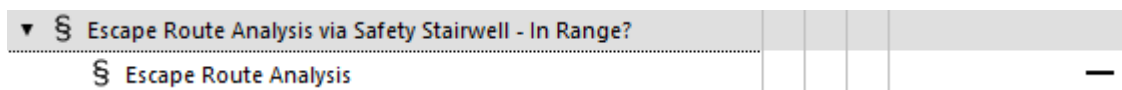


Figure 6.17: Checking Chain Failed in Second Link and Rest Disappears

Code compliance checks, especially complex ones, often require specific modeling requirements. In addition to missing test documentations as discussed above, the overall modeling process requires a more standardized manner. Detailed information regarding certain modeling use cases leads to a more reliable semantic basis of a building model and increases the applicability range of code compliance checking procedures. This standardization, e.g., regarding modeling requirements of utilization units and sub-spaces, can prevent situations of 6.16.

The enriched information for the automated compliance check requires a large amount of manual work. An example of this is the assignment of flammability and fire resistance value to necessary stair components. The flammability of a component results from the relating materials that can be mapped to the entity. In this work, only the assignment of properties was verified in the data quality but missed an additional validation if the model creator correctly assigns these properties due to further fulfilled requirements. Another example is the presence of a safety stairwell which is validated by accessing spaces representing the stairwell and holding the *SafetyStairwell* property. An additional check verifying if the requirements for a safety stairwell are fulfilled would improve the overall reliability of enriched data and decreases the dependence of plain mapped data.

## 6.6 Summary

This chapter examines the concepts of chapter 5 concerning validation of the fifth section of MBO by translating essential information into computer interpretable information. Therefore, already implemented compliance tests of Solibri Office are utilized and extended by six custom implemented tests. The abidance of exchange requirements is validated via three individual tests. The aggregation of building elements is checked by custom implemented check via API and two additional tests provided by Solibri Office are utilized to check the allocation to the correct IFC classes and correctness of enriched attributes. A test building model containing several errors regarding the §33 and §34 of MBO is examined in terms of the model's design quality. Both use cases are validated via seven compliance checks. Five of these are implemented via the Solibri API and validate the existence of escape windows or safety stairwells, and requirements concern necessary stair entities. These testing results of the underlying compliance scenarios and the implementation process are reviewed and discussed concerning their overall strength and limitations. This work outlines the strength of incremental test scenarios utilizing hard-coded tests combined with detailed test documentation to reduce its black-box character and enhance user involvement.



## Chapter 7

# Conclusion

This chapter summarizes the content and results of the underlying work. In addition, the second section gives an outlook based on the evaluated implementation problems.

### 7.1 Summary

The process of digitization transformed all economic sectors and drastically increased its efficiency and overall way of working. The **AEC** sector is one of the least digitized sectors and therefore, misses a lot of potential to face upcoming challenges. The **BIM** method is part of this digitization process and facilitates the use of digital building models throughout the planning, building and maintenance phase.

This work aims to develop an approach to analyze norms and other regulatory texts to be able to identify essential information as exchange requirements for the compliance checking process. These information are then utilized for an automated compliance check concerning the underlying requirements. Therefore, the information has to be transformed into machine-interpretable data and assigned to relating building components in the digital building model.

As a starting point, this work provides fundamental information concerning the open data format **IFC**. An understanding of its overall class structure, inheritance hierarchy and object relationship are essential to guarantee a correct utilization of modelling and export processes. Beside an open data format, **BIM** requires standardized processes to structure new working procedures and large amount of data of underlying projects. For this reason, terms of **IDM**, **MVD** and **LOD** are introduced and its role in a **BIM** project is discussed.

Maintaining a high model quality throughout its life time plays a vital role in the application of **BIM**. **ACCC** describes the process of (semi-) automated software programs, testing a digital building model for its data and design quality, and the overall consistency. These three introduced levels of model quality differ in its characteristics and can strongly influence each other. Code compliance checking helps to identify errors in a building model to communicate them with relating person in charge to enable their elimination in a modelling software later on. The process of **ACCC** face a variety of different fields of application and use cases. Therefore, several technical approaches are developed in the last years to counteract and compensate limitations of existing methods. A standard and most wide spread approach represents hard-coded compliance tests, but which are accompanied by a strong black-box character. This work outlines two additional

approaches: domain specific programming languages by the example of [BERA](#) and a visual programming language, [VCCL](#).

The highest level of model quality represents the design quality claimed by norms and regulatory texts. The [MBO](#) ensures a standardization in the building regulations of federal states in Germany. [MBO](#) itself does not have a legal status but states general claims e.g., fire protection. The core of this work represents the fifth section of [MBO](#) regulating requirements for escape ways, its escape way elements and overall presence of barriers in a building. Content and requirements of [MBO](#) are predominantly represented in a performance-based way due to its general character in legislation. The large amount of norms and regulations and its different representation types resulted in different approaches to automatically extract information and requirements. This work outlines two approaches to enable an automated analysis and structuring of regulatory texts.

For the conceptual implementation of identifying essential information for compliance checking, the RASE mark up technique is manually utilized. It facilitates a well structured text, highlighting essential applying objects, requirements and exceptions. These identified information must be distinguished from standard included data in a building model exported in [IFC](#) to prevent multiple mapping of similar information. Therefore, essential basic functionalities covered by [IFC](#) data scheme are outlined, comprising three dimensional aggregations, aggregation of specific building elements, allocation of underlying building elements to underlying [IFC](#) entities and additional semantic assignment operations like classifications and grouping of building elements. Due to the general character of [MBO](#) it often states vague requirements concerning particular scenarios. To make this more tangible, the term and levels of granularity are introduced and different handling concepts are discussed on the basis of certain examples of the [MBO](#). To be able to enrich digital building models with the identified information, data mapping tables are utilized comprising the underlying information, data types and [IFC](#) entities. Additionally, the compliance checking processes of incremental and full-automated tests are discussed regarding its black-box character and the resulting problems. A technical implementation of code compliance checking tests requires certain functionalities of filtering building element components, querying its properties and geometrical and spatial relations. These functionalities are core methods in every code compliance checking approach and are reviewed according to its technical implementation.

As a proof of concept, the mapping process of all 42 identified information covered by the mapping table is outlined and its technical implementation is reviewed. The test building model is modelled in Revit and enriched via shared parameter files. Required information are then mapped to relating entities in Revit, and exported into [IFC 4](#). This export of additional information is facilitated by user defined property sets which are attached to the underlying [IFC](#) entities. The quality of exported and mapped data was validated in Solibri Office. Custom implemented and already existing tests are used to verify abundance of determined exchange requirements. This process represents an essential part of the workflow and its utilization enables a consistent building model. To validate errors in the building model, the first two paragraphs of the fifth section are employed as use cases.

§33 claims requirements concerning the first and second escape route, and represents the central part for the remaining paragraphs. The following paragraphs, like the second use case §34, represents parts of the escape route and comprise detailed requirements regarding its designs. §34 cover claims concerning necessary stairs of a building and its demands regarding certain properties, like flammability, or design aspects. The first checking scenario comprises three individual tests and can validate §33 except for escape route leading over ramps or ladders, and overall special buildings like schools are not covered. Checking scenario for §34 consists of four individual tests, although the last three sections of §34 in MBO do not cover any specific requirements but can be validated.

All custom implemented compliance tests are described in a test documentation in terms of its functionalities and limitations to reduce the black box character of hard coded tests. Additionally, the checking scenarios consist of highly incremental tests to ensure small and simpler tests, increasing chance of responsible utilization by the user. Nevertheless, applied tests of Solibri Office only provide minimal documentation of overall functionality although its high complexity of escape route calculation and modelling requirements for correct processing. These complex compliance tests outlined the necessity for an increasing standardization of modelling processes to ensure a reliable data basis for code compliance checking. Additionally, the validated compliance results of this work only represents verification of enriched information. This process requires extensions of further tests checking for particular design requirements and thus attesting assigned information. Nevertheless, this work outlined the strength of incremental test scenarios utilizing hard-coded tests combined with detailed test documentation to reduce its black-box character and enhance user involvement. Additionally, the mapping of additional attributes facilitated an automatic validation of the MBO's content and improved the issue management of building models by further automation.

## 7.2 Future Work

This work outlined and evaluated the developed approach for handling and analysing regulatory texts, to identify essential data and enabling a code compliance check regarding these requirements.

This checking process is merely based on querying assigned information. The mapped information can be verified regarding its existence and assignment to correct building elements, but a validation of its overall correctness, representing particular design criteria or other objects, is missing. Therefore, checking scenarios have to be extended by further tests which verify additional properties or design requirements. This extension would represent another security layer and act as an additional validation instance regarding the mapped data. An examples for this is the assignment of flammability values which are as well connected to assigned building materials. This specific process requires extensive data basis covering numerous materials, but can act as another compliance instance to ensure consistent digital building models. Another additional verification off a properties

validness would be e.g., to validate if in front of a *FireExit* window is enough space for rescue via fire brigade.

The enrichment of data and its utilization in processes like code compliance checking depends on adherence regarding specific nomenclature. *bSI'* tool of *bsDD* enables an implementation of property sets and a coherent naming convention of underlying properties resulting in a higher standardization. Additionally, an increasing standardization of modelling processes facilitates a reliable data base and data structure concerning underlying digital building models. This resulting stable and consistent data base could accelerate the expansion of code compliance checking tests covering numerous use cases. Organisations like standards committees can outline general modelling guidelines which are taught in universities or vocational schools and are extended over time.

The examined *MBO* section claimed requirements regarding fire protection of buildings. The mapped information regarding fire protection topics could be utilized by other project stakeholders for their occurring *BIM* workflows. Huge amount of data and multiple assignment of same information can be reduced by examining overlapping of data between different stakeholders. This increased efficiency can be used in similar *BIM* use cases and results in a decreasing risk of errors and a reduced workload.

Solibri released a new add-on to validate building models regarding the *MBO*. It comprises several tests to evaluate the paragraphs of the fifth section in combination with a set of classifications to classify all underlying building elements. The additional attributes outlined in this thesis can be utilized for an automated building element classification via the provided add-on. The official release is one day before the submission date of this thesis. Therefore, an investigation of the determined attributes in combination with the add-on must be conducted in future works.

# Appendix A

The attachment includes in digital form:

- MBO\_AdditionalIdentifiedData
  - Mapping\_Table
  - MBO\_RASE\_MarkUp
  - MBO\_DataTags
  - Mark\_Up\_Table
- Models
  - Example Models (.rvt + .ifc)
  - Shared Parameter File (.txt)
  - User Defined Property Sets (.txt)
  - Solibri File (.smc)
- Solibri Tests
  - Test Documentation
  - Solibri Tests (.jar)
  - Source Code Solibri Tests
  - Solibri Rule Sets (.cset)

# List of Figures

1.1	Manual Error Detection vs. BIM Based Compliance Checking . . . . .	2
2.1	Four Layers of IFC Schema (“buildingSMARTb”, 2021) . . . . .	5
2.2	Most Important Entities of the Inheritance Hierarchy in the IFC Schema (BORRMANN et al., 2015) . . . . .	7
2.3	Scope of Design Transfer View and Reference View (IFC 4) Compared to Coordination View (IFC 2x3) (BALDWIN, 2017) . . . . .	11
2.4	mvdXML Elements (Inspired by CHIPMAN et al., 2016) . . . . .	12
2.5	Level of Development Illustrated by the Example of a Steel Beam (BORRMANN et al., 2015) . . . . .	14
3.1	General Structure of Compliance Checking (PREIDEL and BORRMANN, 2015)	17
3.2	Model’s Quality Levels (PREIDEL, 2020) . . . . .	18
3.3	Schematic Diagram of Black-Box and White-Box Processes (PREIDEL, 2020)	19
3.4	Form Finding and Static Analysis of a Bridge via Kiwi3d (Plug-In for Grasshopper) “TUM_ST”, 2021 . . . . .	23
3.5	VCCL Graph Describing the Access to a Set of Walls. The Blue Trapezoid Represents the Get Access Method and Grey Rectangles Underlying Objects. (PREIDEL and BORRMANN, 2015) . . . . .	24
3.6	VCCL Graph Describing the Central Regulation of DIN 18232-2:2007-11 (PREIDEL and BORRMANN, 2015) . . . . .	25
4.1	Underlying RASE Mark-Up Tags Applied on Standard NS 11001-1 (HJELSETH and NISBET, 2011) . . . . .	32
4.2	NLP Position in Computer Science and Linguistics, and its Possible Technical Implementations . . . . .	33
5.1	Mark-Up Technique Based on the RASE concept . . . . .	38
5.2	An Example of the Hierarchical Aggregation of Spatial Structure Elements and Assignment of Building Elements . . . . .	40
5.3	Aggregation of Spatial Structure Elements . . . . .	41

5.4	Absolute Positioning of <i>IfcSite</i> . . . . .	41
5.5	Linking of Building Components to Spatial Elements . . . . .	42
5.6	Virtual and Physical Space Boundary . . . . .	43
5.7	Object's Shape Representation of <i>IfcStair</i> as an Assembly of Building Elements . . . . .	44
5.8	Classification of Shear Walls with a User Defined Classification System . . . . .	46
5.9	Grouping of two Walls in a <i>Test</i> Group . . . . .	47
5.10	Entity Inheritance of <i>IfcBuildingElement</i> and its Sub-Classes ("buildingSMARTI", 2021) . . . . .	48
5.11	Three Fundamental Element Hierarchy for Information Translation into Computer Readable Data . . . . .	50
5.12	Standard Door Properties Included in Entity Definition ("buildingSMARTm", 2021) . . . . .	55
5.13	Extraction of Additional Door Information of Pset_DoorCommon ("buildingSMARTm", 2021) . . . . .	55
5.14	Extraction of Customer Defined Property Sets for Additional MBO Information . . . . .	56
5.15	Incremental vs "Full-"Test Approach Testing for Compliance with 5th Section of MBO. Incremental Requirements are Divided into Further Sub-Tests . . . . .	58
5.16	Filtering for Component Type Wall to Receive List of Wall Components for Further Processing and Calculation . . . . .	60
5.17	Wall's Property Query regarding its Cladding Thickness, Combustibility and External Positioning . . . . .	61
5.18	Schematic Representation of Topological Operators (PREIDEL, 2020) . . . . .	62
6.1	Underlying Tests Scenarios. Dotted Blocks Represent Custom Implemented Tests. Continuous Blocks Delivered from Solibri Office . . . . .	64
6.2	Underlying Tests of Data Quality Compliance Scenario . . . . .	67
6.3	Adjustment of Filter in Element Aggregation Test to Generate Proper Results . . . . .	68
6.4	Parameter Set Up for <i>Required Property Sets - SOL/203/2.4</i> . . . . .	69
6.5	Test Scenario of Common Rooms. Continuous Blocks Represent Tests Provided by Solibri. Dotted Blocks Represent Custom Implemented Tests via API . . . . .	72
6.6	Adjustment of Gatekeeper Rule Component Handover . . . . .	73
6.7	Test Scenario to Test for Compliance with Paragraph 33 of MBO . . . . .	74

6.8	Test Scenario to Test for Compliance with Paragraph 34 of MBO . . . . .	77
6.9	Results of Data Quality Scenario for Following Compliance Check of §33 and §34 . . . . .	78
6.10	Result Overview of Test Scenario for Common Rooms Checking for First and Second Escape Route . . . . .	79
6.11	Visualization of Common Rooms missing Second Escape Route and Identi- fied Stairwell . . . . .	80
6.12	Test Validates New Assigned Safety Stairwell and No Second Escape Route Is Required . . . . .	81
6.13	Result Overview of Test Scenario for Stairs . . . . .	81
6.14	Visualization of Prohibited Stair Types and Elevator as Escape Route Parts in Building Class 4 . . . . .	82
6.15	Test Validates New Assigned Property Values Concerning Flammability and Types of Stairs . . . . .	82
6.16	Left Example: Represents Overall Utilization Unit and Includes its Sub- Spaces e.g., Living Room. Right Example: Model Only Contains Sub- Spaces and Generates Correct Escape Route . . . . .	83
6.17	Checking Chain Failed in Second Link and Rest Disappears . . . . .	84



# List of Tables

4.1	Examples of POS Tags to Enable Classification of Tokens (ZHANG and EL-GOHARY, 2015) . . . . .	35
5.1	Building Elements and relating Element Types in IFC . . . . .	49

# List of Algorithms

2.1	lfcRoot EXPRESS Specification (“buildingSMARTc”, 2021) . . . . .	8
2.2	Extraction of mvdXML File. Definition of <i>Templates</i> Body (CHIPMAN et al., 2016) . . . . .	13
3.1	Definiton of myrule in BERA (LEE, 2011) . . . . .	22

# References

- BALDWIN, M. (2017). *Der bim-manager - praktische anleitung für das bim-projektmanagement*. Beuth.
- Bim4infra* [lastly accessed: 2021-05-06]. (2020). [https://bim4infra.de/wp-content/uploads/2019/07/BIM4INFRA2020\\_AP4\\_Teil7.pdf](https://bim4infra.de/wp-content/uploads/2019/07/BIM4INFRA2020_AP4_Teil7.pdf)
- BITTNER, T., & SMITH, B. (2002). *A unified theory of granularity, vagueness and approximation*. Department of Computer Science. [https://www.researchgate.net/publication/2498722\\_A\\_Unified\\_Theory\\_of\\_Granularity\\_Vagueness\\_and\\_Approximation](https://www.researchgate.net/publication/2498722_A_Unified_Theory_of_Granularity_Vagueness_and_Approximation)
- BLIND, K., JUNGMITTAG, A., & MANGELSDORF, A. (2021). *Din* [lastly accessed: 2021-01-04]. <https://www.din.de/blob/79542/946e70a818ebdaacce9705652a052b25/gesamtwirtschaftlicher-nutzen-der-normung-data.pdf>
- BORRMANN, A. (2007). *Computerunterstützung verteilt-kooperativer bauplanung durch integration interaktiver simulationen und räumlicher datenbanken*. <https://mediatum.ub.tum.de/doc/618188/618188.pdf>
- BORRMANN, A., KOCH, C., KÖNIG, M., & BEETZ, J. (2015). *Building information modeling - technologische grundlagen und industrielle praxis*. Springer.
- Buildingsmarta* [lastly accessed: 2020-11-27]. (2021). <https://technical.buildingsmart.org/standards/ifc>
- Buildingsmartb* [lastly accessed: 2020-12-01]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/link/introduction.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/link/introduction.htm)
- Buildingsmartc* [lastly accessed: 2020-12-06]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifckernel/lexical/ifcroot.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifckernel/lexical/ifcroot.htm)
- Buildingsmartd* [lastly accessed: 2020-12-07]. (2021). <https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/>
- Buildingsmarte* [lastly accessed: 2021-03-17]. (2021). <https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD1/HTML/schema/ifcproductextension/lexical/ifcspatialstructureelement.htm>
- Buildingsmartf* [lastly accessed: 2021-03-17]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/)
- Buildingsmartg* [lastly accessed: 2021-03-22]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifcproductextension/lexical/ifcrelspaceboundary.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcproductextension/lexical/ifcrelspaceboundary.htm)
- Buildingsmarth* [lastly accessed: 2021-03-22]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcmember.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcmember.htm)
- Buildingsmarti* [lastly accessed: 2021-03-23]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcstair.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcstair.htm)
- Buildingsmartj* [lastly accessed: 2021-03-23]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifckernel/lexical/ifcgroup.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifckernel/lexical/ifcgroup.htm)

- Buildingsmartk* [lastly accessed: 2021-03-23]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4\\_1/FINAL/HTML/schema/ifcexternalreferencerresource/lexical/ifcclassification.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4_1/FINAL/HTML/schema/ifcexternalreferencerresource/lexical/ifcclassification.htm)
- Buildingsmartl* [lastly accessed: 2021-03-25]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifcproductextension/lexical/ifcbuildingelement.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcproductextension/lexical/ifcbuildingelement.htm)
- Buildingsmartm* [lastly accessed: 2021-04-12]. (2021). [https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2\\_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcdoor.htm](https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/schema/ifcsharedbldgelements/lexical/ifcdoor.htm)
- CHIPMAN, T., LIEBICH, T., & WEISE, M. (2016). Mvdxml version 1.1 final specification of a standardized format to define and exchange model view definitions with exchange requirements and validation rules.
- Dibt* [lastly accessed: 2021-01-04]. (2021). <https://web.archive.org/web/20170811104734/https://www.dibt.de/de/Zulassungen/abZ-FAQ-Frage-5.html>
- Din 820-2* [lastly accessed: 2021-02-25]. (2021). <https://www.beuth.de/de/norm/din-820-2/165417565>
- EASTMAN, C., LEE, J.-m., JEONG, Y.-s., & LEE, J.-k. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), 1011–1033. <https://doi.org/https://doi.org/10.1016/j.autcon.2009.07.002>
- FIEDLER, J. (2015). *Modernization scenarios of the building permit procedure in consideration of new technological tools* [lastly accessed: 2021-05-16]. <https://repositum.tuwien.at/handle/20.500.12708/2830>
- HJELSETH, E., & NISBET, N. (2011). *Capturing normative constraints by use of the semantic mark-up rase methodology*. Department of Mathematical Sciences; Technology. [https://www.researchgate.net/publication/265059517\\_CAPTURING\\_NORMATIVE\\_CONSTRAINTS\\_BY\\_USE\\_OF\\_THE\\_SEMANTIC\\_MARK-UP\\_RASE\\_METHODODOLOGY](https://www.researchgate.net/publication/265059517_CAPTURING_NORMATIVE_CONSTRAINTS_BY_USE_OF_THE_SEMANTIC_MARK-UP_RASE_METHODODOLOGY)
- HUDECZEK, D. (2017). *Formalisierung von normen mithilfe von auszeichnungssprachen für die automatisierte konformitätsüberprüfung*. [https://publications.cms.bgu.tum.de/theses/2017\\_Hudeczek.pdf](https://publications.cms.bgu.tum.de/theses/2017_Hudeczek.pdf)
- JONES, R., & HOWARTH, A. (2019). *World-gbc* [lastly accessed: 2021-01-07]. <https://www.worldgbc.org/news-media/WorldGBC-embodied-carbon-report-published>
- LEE, J. (2011). Building environment rule and analysis (bera) language and its application for evaluating building circulation and spatial program. *Advanced Engineering Informatics*. <https://smartech.gatech.edu/handle/1853/39482>
- MATTIUZZO, C., & MIESNER, S. (2021). *Genormte normensprache* [lastly accessed: 2021-02-25]. <https://www.kan.de/publikationen/kanbrief/normatives-und-informatives/genormte-normensprache/>
- MAYR, J. (2014). *Anforderungen, planung und ausführung von rettungswegen*. Feuertrutz Verlag. [https://www.kalksandstein.de/bv\\_ksi/binaries/content/83044/file\\_bauseminare\\_ks-ost\\_2014\\_vortrag\\_josef\\_mayr\\_de.pdf](https://www.kalksandstein.de/bv_ksi/binaries/content/83044/file_bauseminare_ks-ost_2014_vortrag_josef_mayr_de.pdf)
- Mckinsey* [lastly accessed: 2021-01-02]. (2016). <https://www.mckinsey.com/business-functions/operations/our-insights/imagining-constructions-digital-future>

- Musterbauordnung* [lastly accessed: 2021-01-06]. (2019). <https://www.bauministerkonferenz.de/Dokumente/42323097.pdf>
- PLUM, A. (2016). *Brandlasten in rettungswegen, grundlagen für einzelfallbetrachtungen*. BFT Cognos GmbH. [https://www.bft-cognos.de/assets/files/Veroeffentlichungen/10\\_Brandlasten%20in%20Rettungswegen.pdf](https://www.bft-cognos.de/assets/files/Veroeffentlichungen/10_Brandlasten%20in%20Rettungswegen.pdf)
- POPGAVRILOVA, G. (2020). *Assuring building information quality for building analytics by translating use cases of bim@sre standard into the mvd format*. [https://publications.cms.bgu.tum.de/theses/2020\\_Popgavrilova\\_Braun\\_MVD.pdf](https://publications.cms.bgu.tum.de/theses/2020_Popgavrilova_Braun_MVD.pdf)
- PREIDEL, C. (2020). *Automatisierte konformitätsprüfung digitaler bauwerksmodelle hinsichtlich geltender normen und richtlinien mit hilfe einer visuellen programmiersprache* [lastly accessed: 2021-01-04]. <https://mediatum.ub.tum.de/doc/1534486/1534486.pdf>
- PREIDEL, C., & BORRMANN, A. (2015). *Automated code compliance checking based on a visual language and building information modeling*. [https://publications.cms.bgu.tum.de/2015\\_Preidel\\_ISARC.pdf](https://publications.cms.bgu.tum.de/2015_Preidel_ISARC.pdf)
- RESHAMWALA, A., MISHRA, D., & PAWAR, P. (2013). Review on natural language processing.
- RIBEIRINHO, M., MISCHKE, J., STRUBE, G., SJÖDIN, E., BLANKCO, J., PALTER, R., BJÖRCK, E., ROCKHILL, D., & ANDERSSON, T. (2020). *The next normal in construction* [lastly accessed: 2021-07-09]. <https://www.mckinsey.de/~ /media/mckinsey/business%5C%20functions/operations/our%5C%20insights/the%5C%20next%5C%20normal%5C%20in%5C%20construction/the-next-normal-in-construction.pdf>
- Roland berger focus* [lastly accessed: 2021-01-06]. (2017). [https://www.rolandberger.com/de/Insights/Publications/Point-of-View-Details\\_29440.html](https://www.rolandberger.com/de/Insights/Publications/Point-of-View-Details_29440.html)
- SOLIHIN, W., & EASTMAN, C. (2015). *Automation in construction*.
- SOLIHIN, W., EASTMAN, C., & LEE, Y. C. (2016). A framework for fully integrated building information models in a federated environment. *Advanced Engineering Informatics*, 30(2), 168–189. <https://doi.org/https://doi.org/10.1016/j.aei.2016.02.007>
- Technical roadmap 2020*. (2021). [https://buildingsmart-1xbd3ajdayi.netdna-ssl.com/wp-content/uploads/2020/09/20200430\\_buildingSMART\\_Technical\\_Roadmap.pdf](https://buildingsmart-1xbd3ajdayi.netdna-ssl.com/wp-content/uploads/2020/09/20200430_buildingSMART_Technical_Roadmap.pdf)
- Technical roadmap 2021* [lastly accessed: 2021-07-09]. (2021). <https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/>
- Tum\_st* [lastly accessed: 2021-05-19]. (2021). <https://www.bgu.tum.de/st/software/forschung/kiwi3d/>
- Vdi* [lastly accessed: 2021-01-04]. (2021). <https://www.vdi.de/ueber-uns/organisation>
- ZHANG, J., & EL-GOHARY, N. (2015). Semantic nlp-based information extraction from construction regulatory documents for automated compliance checking. *ASCE*. [https://www.researchgate.net/publication/273024453\\_Semantic\\_NLP-Based\\_Information\\_Extraction\\_from\\_Construction\\_Regulatory\\_Documents\\_for\\_Automated\\_Compliance\\_Checking](https://www.researchgate.net/publication/273024453_Semantic_NLP-Based_Information_Extraction_from_Construction_Regulatory_Documents_for_Automated_Compliance_Checking)

## **Appendix B**

### **Declaration**

I hereby affirm that I have independently written the thesis submitted by me and have not used any sources or aids other than those indicated.

---

Munich, July 15, 2021, Sebastian Schliski