

DFBI 2025 aims to encourage the international exchange of innovative ideas between researchers from academia and industry. In addition to knowledge dissemination, the conference offers a valuable platform for professional networking, particularly benefiting university professors, graduate students, and postdoctoral researchers.

Research Article

LLM-Based IFC Mapping and Automatic IDS Generation for Supporting Automated Code Compliance

Sherief Ali, Markus König

Chair of Computing in Engineering, Faculty of Civil and Environmental Engineering, Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum, Germany

Correspondence: Sherief.ali@ruhr-uni-bochum.de

Copyright: Copyright: © 2025 by the authors.

DFBI is an open-access proceedings distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0). View this license's legal deed at <https://creativecommons.org/licenses/by/4.0/>



Abstract

Building regulations are typically expressed in natural language, which makes their interpretation heavily dependent on the manual work of domain experts. Practitioners must read regulatory clauses, identify the relevant requirements, and convert them into rule sets for BIM-based compliance checking. This process is time-consuming, prone to errors, and difficult to scale across different regulatory domains. Although recent research has begun to automate parts of this workflow, the subsequent steps of integrating regulatory information into BIM processes and generating machine-readable rules still rely largely on manual effort. This paper introduces a two-stage approach that automates these remaining tasks using large language models (LLMs) through the Gemini API. The method operates on regulatory clauses that have already been semantically tagged with objects, properties, qualities, and values, providing a structured representation of the text from which the two stages proceed. In the first stage, the LLM performs schema-aware reasoning over a knowledge base of Industry Foundation Classes (IFC) 4.3 definitions to map each tagged element to its corresponding IFC class, property set, or attribute. In the second stage, these mappings are used to automatically generate Information Delivery Specifications (IDS) that encode the relevant attribute-based regulatory requirements. The workflow is validated using clauses from the International Building Code (IBC), with the generated IDS executed in Solibri to assess correctness and practical applicability. The results demonstrate that LLM-driven semantic mapping, combined with automated IDS generation, significantly enhances the consistency and scalability of digital compliance processes, providing a practical approach to integrating regulatory knowledge into openBIM environments.

Keywords: Code Compliance; Industry Foundation Classes; Information Delivery Specifications; Building Information Modeling

Highlights

- Deep learning extracts key compliance information from regulatory clauses.
- Language models align regulatory meaning with IFC schema concepts for BIM use.
- Automated IDS generation enables machine-checkable requirements in openBIM workflows.

Cite this article:
Lastname, F. M., & Lastname, F. M. (Year). Title of the article. Title of the Journal, Volume (Issue), Page range. <https://doi.org/10.66408/dfbi.2026.26308>

1 Introduction

The Architecture, Engineering, and Construction (AEC) industry is continuously changing, driven by new technologies, tighter sustainability targets, and increasing demands for quality and transparency. In this context, regulatory documents provide the binding framework that guides planning, approval, design, construction, and operation. They are essential to ensure that buildings and infrastructure meet safety, environmental, accessibility, and performance requirements. Yet, working with regulations in daily practice is still largely manual: experts must read long texts, identify relevant clauses, interpret legal language, and translate requirements into checks and deliverables. This manual extraction and interpretation is time-consuming, difficult to scale, and can lead to inconsistent outcomes across projects or even across different reviewers (Eastman et al., 2009) (Borrmann et al., 2018).

Automatic Code Compliance (ACC), therefore, remains a key research topic, aiming to verify digital building information models against regulatory requirements in a reliable and repeatable way. Existing approaches range from manual and semi-automated checking to more advanced workflows that combine structured rule languages, model checking systems, and data-driven methods (Fuchs et al., 2022). Rule-based ACC pipelines are effective when rules are already formalized, but they typically require substantial expert effort to convert natural language clauses into executable logic. This translation step is a major bottleneck, and it is prone to errors caused by ambiguous phrasing, complex legal terminology, and inconsistent rule interpretations (Eastman et al., 2009). In contrast, machine learning approaches reduce the dependence on fully manual rule encoding by learning how regulations are phrased and which parts of a clause carry actionable compliance information. By emphasizing semantic understanding of regulatory text, these methods can support more consistent extraction of relevant constraints and reduce variability in interpretation (Alnuzha & Bloch, 2025).

Building on this direction, this paper presents a workflow that automates the remaining steps after semantic tagging, namely the integration of regulatory information into BIM semantics and the generation of machine-readable checking specifications. The method assumes that regulatory clauses are already semantically tagged with Object, Property, Quality, and Value, providing a structured representation of the requirement content. Based on this input, the workflow applies a two-stage LLM approach via the Gemini API. First, the LLM performs schema-aware mapping by aligning the tagged elements to corresponding IFC 4.3 classes, property sets, and attributes. Second, these mappings are used to automatically generate IDS that encode attribute-based regulatory requirements for openBIM validation. The workflow is validated using clauses from the International Building Code (IBC) and executed in Solibri to assess correctness and practical applicability. Overall, the approach reduces manual effort in rule authoring and improves the consistency and scalability of BIM-based compliance processes (Fuchs et al., 2022; Alnuzha & Bloch, 2025).

2 Related Work

Regulatory compliance checking has been studied for decades, but the main bottleneck remains the “translation gap” between regulations written for human interpretation and formal rules that software can execute. ACC research, therefore, increasingly focuses on methods that can interpret regulatory language more consistently and convert it into structured, machine-processable representations. Within this line of work, deep learning and modern NLP have become central because they can capture linguistic patterns and domain-specific phrasing without requiring every clause to be manually encoded into logic.

ACC is explored in various research studies, employing a range of deep learning models. Different approaches are designed to interpret the natural language of building codes and to capture the semantics of sentences. Natural Language Processing (NLP) techniques are crucial in these efforts, allowing systems to process and understand regulatory texts. For instance, Named Entity Recognition (NER), which identifies and classifies entities within text, and Part-of-Speech (POS) tagging, which assigns parts of speech to words to better understand their grammatical roles (Manning et al., 2014). Zhang and El-Gohary (2022) proposed a deep learning-based approach to generate intelligent building codes. They introduced a semantic representation called the Multi-Form Semantic (MFS) requirement hierarchy. This hierarchy organises requirements into units and their interrelations, with each unit further defined by Semantic Information Elements (SIEs) such as subject, compliance attributes, and quantitative details. An intelligent code is defined as natural-language requirements linked to their hierarchies. The authors trained a Recurrent Neural Network (RNN) on 7,500 annotated sentences and tested it on 600 manually created sentences to generate coherent intelligent building-code requirements by feeding requirement units into the model. Other studies focus on relations in the text. For instance, Wang and El-Gohary (2023) proposed a relation extraction method using deep learning to identify relations from unstructured text in construction safety requirements. The authors evaluated their model using two deep learning architectures, which are Convolutional Neural Network (CNN) and RNN and reported that the CNN-based model captured features more precisely. Xue et al. (2024) addressed the common challenge of limited training data by presenting a deep learning and NLP-based question-answering (QA) framework for building codes, consisting of two main components: a retriever and a reader. The retrievers used sparse and dense features, while the readers utilized transformer-based language models fine-tuned to a building-code dataset. Yang and Zhang (2024) integrated Generative Pretrained Transformers (GPT) models with prompt engineering to combine the precision of rule-based systems with the flexibility of machine learning. Using a prompt-based approach, the GPT model can interpret and translate the natural language of building codes into machine-readable formats.

In summary, prior work demonstrates strong progress in structuring regulatory requirements, extracting semantic relations, and improving access to building codes through QA systems or prompt-based translation. However, many approaches focus either on higher-level representations (e.g., requirement hierarchies and relation graphs) or require substantial task-specific prompting and post-processing to obtain outputs that can be reused reliably in BIM-based checking workflows. This paper addresses this gap by focusing on a practical intermediate representation: semantically tagged entities that capture the actionable parts of a clause. Using Object, Property, Quality, and Value as a compact semantic layer, the approach supports consistent alignment to machine-interpretable representations and reduces ambiguity during the transformation of regulatory text into checkable BIM information requirements.

3 Methodology

This study applies a quantitative, experimental research design to evaluate a two-stage automation workflow for translating semantically structured regulatory clauses into machine-checkable requirements for openBIM compliance checking. The workflow assumes that regulatory clauses are already enriched with semantic tags, which are Object, Property, Quality, and Value. These tags provide a structured representation of the requirement content. Building on this structured input, the proposed method automates (1) the integration of regulatory information into BIM semantics through schema-aware IFC mapping, and (2) the generation of machine-readable compliance specifications as IDS. Figure 1 provides an overview of the pipeline from tagged clauses to executable checks in a model-checking environment. The remainder of this section describes the data preparation and preprocessing steps, and then details the implementation of the two automation stages and how the results are validated in practice.

The remainder of this section is organised as follows. Section 3.1 (Data Preparation) describes the datasets and clause sources used in this study, including semantically tagged regulatory clauses and

the example clauses selected for evaluation and validation. It also explains how clause instances are stored to maintain traceability from the original text to the produced IFC mappings and IDS output. Section 3.2 (Data Preprocessing) explains how the tagged entities are extracted and grouped into Object, Property, Quality, and Value, and how this structured representation is converted into stable inputs for the LLM. It also describes the preparation of the IFC 4.3 schema knowledge base used to constrain and verify mapping results. Section 3.3 (Model Development) details the two-stage Gemini-based pipeline: first, schema-aware mapping of tagged entities to IFC classes, property sets, and attributes; second, automatic IDS generation and XML serialisation. Finally, this section describes how the generated IDS is executed in Solibri on IFC4x3 models to evaluate executability and practical alignment with the regulatory intent.

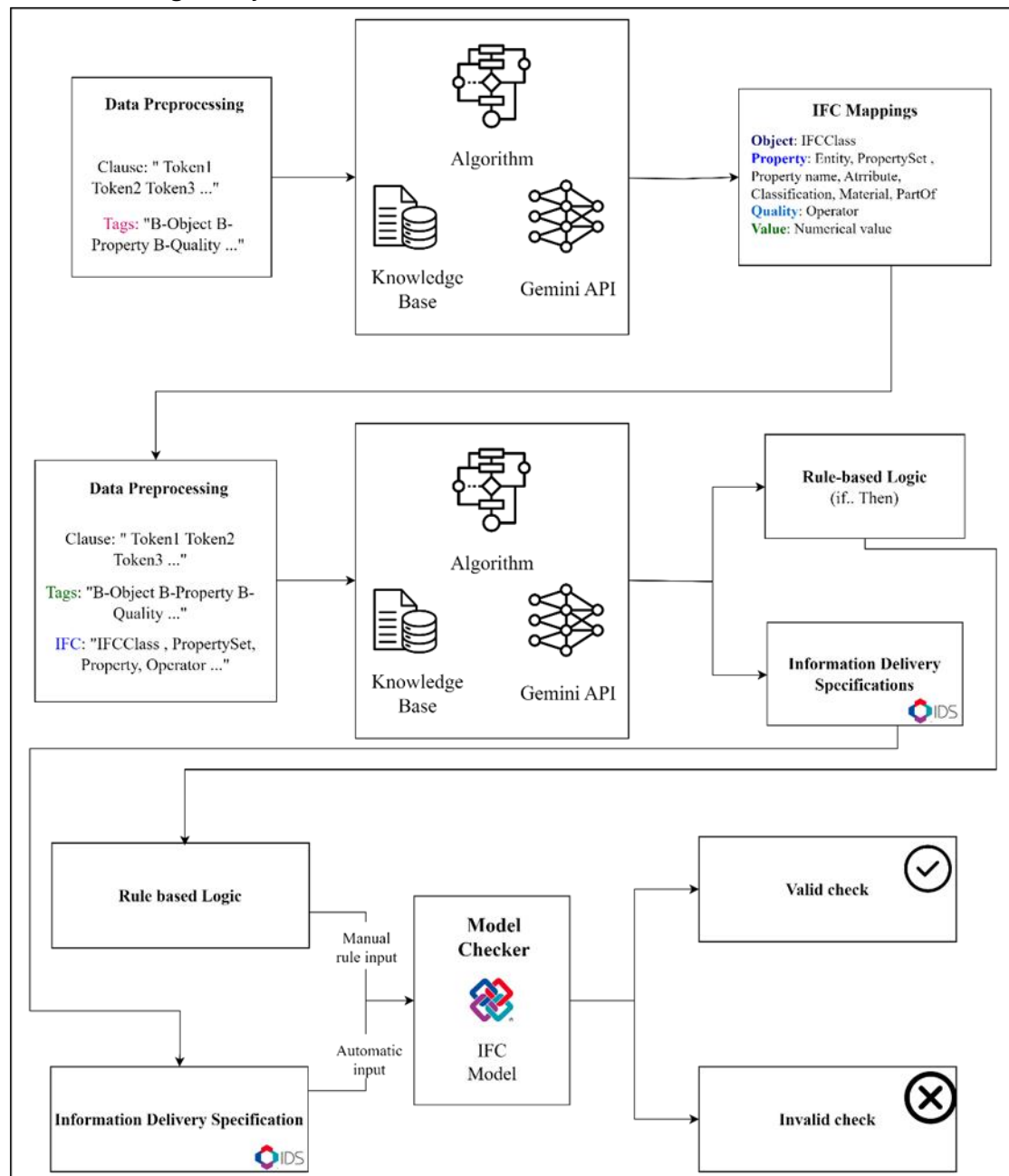


Figure 1. Methodology Design

3.1 Data Preparation

A key challenge in ACC is the lack of publicly available datasets, which could significantly accelerate the development and testing of new methodologies and techniques. As highlighted by (Amor & Dimyadi,

2021), there is no publicly available regulatory dataset that can be used to evaluate the quality of ACC systems. (Hettiarachchi et al., 2024) addressed this issue by introducing the CODE-ACCORD corpus, which contains 862 clauses extracted from building regulations in both England and Finland. The authors manually labelled and annotated a total of 4,297 entities in the CODE-ACCORD corpus.. The dataset is split into training and testing, each containing sentences and their corresponding tags. These sentences are formatted in a structured manner, with both the raw text extracted and associated tags listed in CSV format (Hettiarachchi et al., 2023).

The sentences in the dataset (Hettiarachchi et al., 2023) are annotated using the Begin-Inside-Outside (BIO) tagging scheme (Ramshaw & Marcus, 1995), which is widely used for labelling sequences in tasks such as NER. The BIO tags are applied to every word in the clauses to denote whether a word is the beginning of a specific entity (B-), inside the entity (I-), or outside of any entity (O). For instance, in the phrase "Load-bearing wall", the entity "Load-bearing wall" is annotated as "B-tag I-tag I-tag" because all the words describe the same noun, 'wall', with the description starting from 'load' and ending at 'wall'. However, 'O' denotes any word that does not contribute meaningful information to the sentence

This BIO scheme ensures that the boundaries of multi-word entities, such as legal terms, are marked, allowing for precise entity recognition. The authors (Hettiarachchi et al., 2024) established a set of specific entity categories to guide the tagging of sentences in the CODE-ACCORD corpus, ensuring that every term within the clauses is assigned to one of these predefined categories. This systematic approach helps maintain consistency in labelling and supports the accurate classification of regulatory terms. These entity categories, described in Table 1, include key elements that are central to defining compliance criteria in building regulations.

Table 1. Description of entity categories from the CODE-ACCORD dataset (Hettiarachchi et al., 2023)

Named Entity	Description
Object	An ontological concept that represents a thing that is subject to a particular requirement (e.g., a window, a fire door)
Property	Property of an object (e.g., width, height)
Quality	Unlike properties, qualities are not always quantifiable (e.g., horizontal, self-closing)
Value	Numerical measurement or defined quantity associated with a property (e.g. 1,500 millimetres, five percent)

Based on the entity categories described in Table 1, the sentences in the dataset are annotated according to these specific labels (Hettiarachchi et al., 2024), ensuring that each term is appropriately tagged. This annotation process is illustrated in Table 2, where each word in the sentence is assigned a corresponding BIO tag that reflects its role within the regulatory clause.

Table 2. Examples of annotated clauses in the dataset from the CODE-ACCORD dataset (Hettiarachchi et al., 2023)

Clause	Annotated Clauses
1	The area of windows, roof windows, rooflights and doors should not exceed 25 of the total floor area of the dwelling. The (O) area (B-property) of (O) windows (B-object), (O) roof (B-object) windows (I-object), (O) rooflights (B-object) and (O) doors (B-object) should (O) not (O) exceed (O) 25 (B-value) of (O) the (O) total (B-property) floor (I-property) area (I-property) of (O) the (O) dwelling (B-object). (O)
2	On stairs and ramps, a handrail must be installed for the entire length and on both sides of the structure. On (O) stairs (B-object) and (O) ramps (B-object), (O) a (O) handrail (B-object) must (O) be (O) installed (O) for (O) the (O) entire (B-quality) length (I-quality) and (O) on (O) both (B-quality) sides (I-quality) of (O) the (O) structure (B-object). (O)

3.2 Data Preprocessing

Preprocessing ensures that the tagged clause content can be consumed reliably by the LLM-based stages and that outputs can be transformed into formal IDS structures. First, clause text is normalised (e.g., consistent whitespace, punctuation handling) while preserving the legal meaning. Second, the semantic tags are converted into a structured input format that separates the four entity groups (Object/Property/Quality/Value) and preserves their surface forms as they appear in the clause. This structured representation is used to build prompts that are stable and reproducible across clauses, reducing ambiguity during inference.

In parallel, an IFC 4.3 knowledge base is prepared to support schema-aware reasoning. This knowledge base contains definitions and relationships needed for mapping decisions, such as IFC class names, property set structures, attribute names, and typical associations between element types and properties. For robust post-processing, the workflow also defines normalization routines for outputs (e.g., canonical naming, consistent identifiers, value typing) so that the generated mappings can be deterministically converted into IDS fields.

3.3 Model Development

Model development in this paper focuses on implementing and operationalizing the two LLM-driven automation stages using the Gemini API, rather than training a new language model. The system is designed as a controlled prompting and post-processing pipeline that takes semantically tagged clause inputs and produces two machine-actionable outputs: IFC-semantic mappings and IDS specifications.

In the first stage, the LLM performs schema-aware IFC mapping. Given the tagged entities and access to the IFC 4.3 knowledge base, the model reasons about the most appropriate targets for each tag. Concretely, the stage outputs a structured mapping where:

- **Object** is mapped to an IFC class (e.g., IfcWall, IfcDoor, IfcSpace),
- **Property** is mapped to the most appropriate IFC representation (e.g., attribute, property set + property, or classification reference),
- **Quality** is mapped to a constraint operator or requirement mode,
- **Value** is mapped to a typed threshold (numeric/categorical), including unit handling when applicable.

To improve reliability, the prompt instructs the model to select outputs only from the provided IFC schema vocabulary and to return mappings in a constrained JSON-like structure. Post-processing

validates the returned identifiers against the IFC knowledge base and flags incomplete or inconsistent mappings for review.

In the second stage, the validated IFC mappings are used to automatically generate IDS. The workflow converts the mapping outputs into IDS structures that encode attribute-based requirements: the target IFC class (from Object), the relevant attribute or property definition (from Property), and the constraint logic (from Quality and Value). The generated IDS are serialised into IDS XML so existing openBIM checking tools can directly consume them.

Finally, the workflow is validated by executing the generated IDS in Solibri against IFC models, producing pass/fail outcomes that indicate whether the generated IDS are syntactically valid, executable, and practically aligned with the intended regulatory requirement. The evaluation focuses on the correctness of the IFC mapping outputs, the completeness of the generated IDS, and the practical feasibility of running the checks within a model-checking environment.

4 Results and Analysis

4.1 Model Evaluation

4.1.1 LLM-based Prediction Model

To demonstrate the effectiveness of the proposed workflow, the regulatory clause “The minimum wall thickness for the standard fire resistance EI 30 is 60 mm” was selected from the Eurocode dataset and processed through the full pipeline. This example illustrates how the predicted BIO tags provide a structured input for the LLM-based mapping stage, which then produces IFC-aligned schema mappings that can be used in downstream rule formalisation.

The clause combines two typical compliance aspects that are often handled separately in practice: a quantitative geometric threshold (wall thickness) and a performance requirement (fire resistance rating). The workflow correctly maps the Object (“Wall”) to IfcWall. It further maps the Property related to thickness to a quantity-based IFC representation (Qto_WallBaseQuantities.Width) and normalizes the threshold 60 mm to 0.06 m. In parallel, the pipeline captures the performance-related part of the sentence by mapping “standard fire resistance” to Pset_WallCommon.FireRating and assigning EI 30 as its corresponding value. The constraint direction is also preserved by mapping the keyword “minimum” to the operator $\geq$. Overall, this clause demonstrates that the workflow can integrate dimensional constraints with performance-oriented attributes within a single requirement, while producing structured IFC mappings consistent with the gold standard.

Table 3. . Example for LLM-based IFC element predictions (entity-level mapping vs. gold standard)

Entity Phrase	BIO Tag	Predicted LLM Mapping	Gold Standard
Wall	B-Object	IfcWall	IfcWall
standard fire resistance	B-Property / I-Property / I-Property	Pset_WallCommon.FireRating	Pset_WallCommon.FireRating
EI 30	B-Value	EI 30	EI 30
thickness	I-Property	Qto_WallBaseQuantities.Width	Qto_WallBaseQuantities.Width
minimum	B-Quality	operator: $\geq$	operator: $\geq$
60 mm	B-Value / I-Value	0.06 m	0.06 m

4.1.2 Formalization of Information Requirements

Building on the IFC-aligned outputs from Section 4.1.1, the next step is to formalize the extracted information requirements into a machine-checkable specification. In this paper, this formalisation is performed using Information Delivery Specifications (IDS), which provide a standardised and tool-compatible way to express what information an IFC model must contain. IDS is well-suited for clearly defined, attribute-based requirements (e.g., material, classification, specific property values), but it is not intended to represent complex logical dependencies, spatial reasoning, or multi-step design checks. Therefore, the focus here is on requirements that can be expressed as explicit constraints over IFC classes, attributes, and properties and validated automatically in IDS-compliant checking tools.

```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <ids:ids xmlns:ids="http://standards.buildingsmart.org/IDS" xmlns:xs="http://www.w3.org/2001/XMLSchema"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://standards.buildingsmart.org/IDS
4   http://standards.buildingsmart.org/IDS/1.0/ids.xsd">
5   <ids:info>
6     <ids:title>Minimum Wall Thickness</ids:title>
7     <ids:author>sherief.ali@ruhr-uni-bochum.de</ids:author>
8     <ids:date>2025-09-17</ids:date>
9   </ids:info>
10  <ids:specifications>
11    <ids:specification name="Specification 1" ifcVersion="IFC4X3_ADD2">
12      <ids:applicability>
13        <ids:entity>
14          <ids:name>
15            <ids:simpleValue>IFCWALL</ids:simpleValue>
16          </ids:name>
17        </ids:entity>
18        <ids:property dataType="IFCLABEL">
19          <ids:propertySet>
20            <ids:simpleValue>Pset_WallCommon</ids:simpleValue>
21          </ids:propertySet>
22          <ids:baseName>
23            <ids:simpleValue>FireRating</ids:simpleValue>
24          </ids:baseName>
25          <ids:value>
26            <ids:simpleValue>EI 30</ids:simpleValue> </ids:value>
27        </ids:property>
28      </ids:applicability>
29      <ids:requirements>
30        <ids:property cardinality="required" dataType="IFCLENGTHMEASURE">
31          <ids:propertySet>
32            <ids:simpleValue>Qto_WallBaseQuantities</ids:simpleValue>
33          </ids:propertySet>
34          <ids:baseName>
35            <ids:simpleValue>Width</ids:simpleValue> </ids:baseName>
36          <ids:value>
37            <ids:simpleValue>0.06</ids:simpleValue>
38          </ids:value>
39        </ids:property>
40      </ids:requirements>
41    </ids:specification>
42  </ids:specifications>
43 </ids:ids>

```

Figure 2. Example for IDS Predictions

To illustrate the formalisation step, the same clause used in Section 4.1.1 is reused: “The minimum wall thickness for the standard fire resistance EI 30 is 60 mm.” The IFC mapping results in Table 3 provide the required building blocks for IDS generation: (i) the target element type (IfcWall), (ii) the contextual condition (Pset_WallCommon.FireRating = EI 30), and (iii) the measurable requirement (Qto_WallBaseQuantities.Width ≥ 0.06 m) after unit normalisation. During IDS formulation, these parts are distributed into the two central IDS sections:

- Applicability defines which model elements the rule applies to, here: walls with a declared fire rating of EI 30.
- Requirements define what must be true for those elements, here: the wall thickness must meet the minimum threshold (≥ 0.06 m).

This structuring preserves the intent of the regulation in a format that is directly executable: the applicability condition prevents overgeneralization of the rule to all walls, while the requirements section encodes the quantitative constraint in an IFC-consistent way. The resulting IDS specification can then be imported into validation tools (e.g., Solibri) to perform automated checking against IFC models. Figure 2 illustrates the generated IDS XML for this example, showing how the applicability condition (IfcWall with FireRating = EI 30) and the thickness requirement (Width = 0.06) are encoded in a single IDS specification.

4.2 Model Validation

To validate practical applicability beyond entity-level mapping, the workflow was tested on a representative clause from the International Building Code (IBC) and executed end-to-end in a model-checking environment. The selected clause was:

“Exterior walls separating the exterior area of assisted rescue from the interior of the building shall have a minimum fire-resistance rating of 1 hour.”

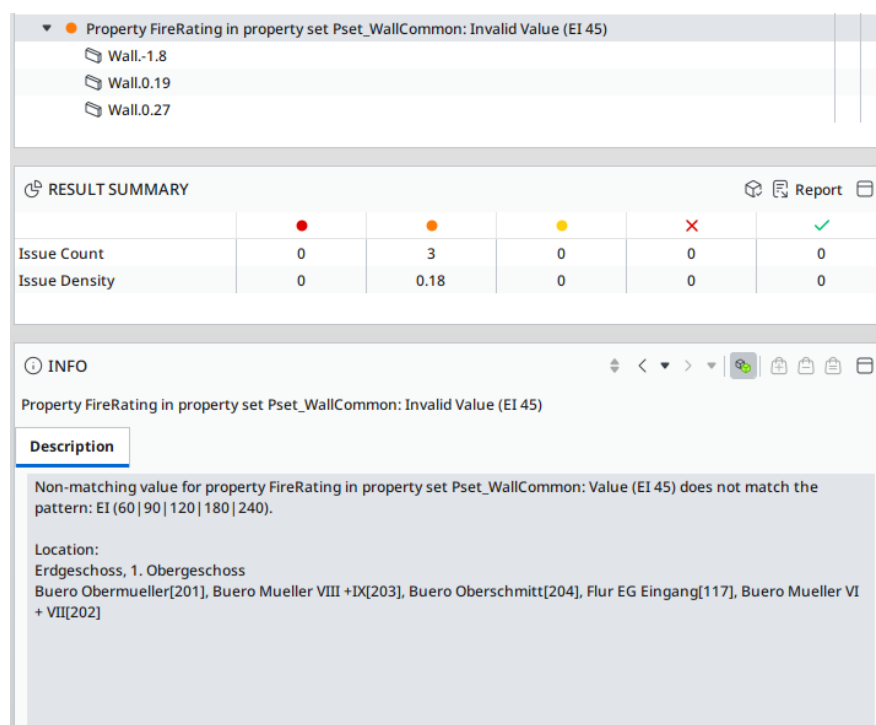


Figure 3. Walls fail the check in Solibri

Based on the semantically tagged clause, the LLM mapped “Exterior walls” to IfcWall and refined the scope using Pset_WallCommon.IsExternal = True to represent the exterior condition. The fire-resistance requirement was mapped to Pset_WallCommon.FireRating, and the value “1 hour” was standardised to EI 60 according to common fire classification notation. Because FireRating is defined as IfcLabel (string) in IFC, the “minimum” logic cannot be encoded using numeric constraints such as minInclusive in IDS. Instead, the requirement was represented using an IDS-compliant pattern

restriction that captures accepted discrete fire rating values (e.g., EI 60, EI 90, EI 120, EI 180, EI 240). This preserves the intent of a minimum threshold while remaining valid under IDS data typing constraints.

The generated IDS was then imported into Solibri and executed on an IFC4x3 project model containing a large set of wall elements. The check results clearly separated compliant from non-compliant exterior walls: 96 walls were reported as failing due to fire rating values outside the accepted pattern (e.g., EI 45 or EI 30), while 13 walls were reported as passing with values within the pattern (e.g., EI 90 and EI 120). Figure 3 provides an example of a failed instance in Solibri, where the checker reports a non-matching value for FireRating in Pset_WallCommon: the wall has EI 45, which does not satisfy the allowed pattern EI (60|90|120|180|240). This illustrates that the generated IDS is not only syntactically valid and executable, but also produces interpretable feedback at the element level, supporting practical model checking within an openBIM workflow.

5 Conclusion

This paper addressed a key bottleneck in Automated Code Compliance: converting natural-language regulatory clauses into structured, machine-checkable requirements for openBIM workflows. A deep-learning entity extraction model was developed by fine-tuning BERT on the CODE-ACCORD corpus using BIO labels across Object, Property, Quality, and Value. The model achieved high precision (0.98), indicating reliable identification of compliance-relevant information in unseen clauses. Building on these structured outputs, a two-stage LLM workflow was applied to (i) align tagged entities with IFC 4.3 schema concepts and (ii) formalize the resulting information requirements into IDS specifications.

The results show that the workflow can combine contextual applicability conditions with measurable requirements in a single machine-readable specification. The Eurocode example demonstrated consistent IFC mappings for both performance descriptors (FireRating) and quantitative thresholds (wall thickness), supporting direct IDS formulation. Validation on an IBC clause further demonstrated practical applicability: the generated IDS was executable in Solibri and successfully distinguished compliant from non-compliant wall elements in an IFC4x3 project model. Regarding the limitations of IDS, the standard is primarily designed for expressing and validating attribute- and property-based information requirements (i.e., whether specific data exists and matches an expected value or pattern). It is less suited for requirements that depend on complex logic, such as multi-condition dependencies across elements, spatial relationships, geometric reasoning, or higher-level performance checks that require rule chaining or contextual interpretation beyond single-property constraints.

Overall, the proposed approach improves scalability and consistency by reducing manual effort in early interpretation and rule authoring, and by producing artifacts that integrate directly with openBIM checking tools. Future work should expand evaluation across more regulatory domains and clause types, and further address constraints that require richer logic than IDS alone can express.

Acknowledgements

The author expresses his sincere appreciation to the Chair of Computing in Engineering for generously funding this research paper. Their support has played a pivotal role in facilitating the study and contributing to advancements in the field. This research was funded by the mFUND research programme of the Federal Ministry for Digital and Transport (BMDV) (funding code: 01F2253B).

Funding

BIM Breitband - mFUND - BMDV - 01F2253B

Data Availability Statement

The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare no conflict of interest.

References

- Alnuzha, M. & Bloch, T. (2025) The role of machine learning in automated code checking – a systematic literature review. *Journal of Information Technology in Construction*, 30, 22–44. Available from: <https://doi.org/10.36680/j.itcon.2025.002>.
- Amor, R. & Dimyadi, J. (2021) The promise of automated compliance checking. *Developments in the Built Environment*, 5, 100039. Available from: <https://doi.org/10.1016/j.dibe.2020.100039>.
- Borrmann, A., König, M., Koch, C. & Beetz, J. (2018) *Building Information Modeling*. Springer.
- buildingSMART (2024) buildingSMART, 2024. Available from: <https://www.buildingsmart.org/>.
- Clemen, C. & Gruendig, L. (2006) *The Industry Foundation Classes (IFC) -- Ready for Indoor Cadastre*.
- Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
- Eastman, C., Lee, J., Jeong, Y. & Lee, J. (2009) Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), 1011–1033. Available from: <https://doi.org/10.1016/j.autcon.2009.07.002>.
- Fuchs, S., Witbrock, M., Dimyadi, J. & Amor, R. (2022) Neural Semantic Parsing of Building Regulations for Compliance Checking. *IOP Conference Series: Earth and Environmental Science*, 1101(9), 92022. Available from: <https://doi.org/10.1088/1755-1315/1101/9/092022>.
- Hettiarachchi, H., Dridi, A., Gaber, M.M. & Parsafard, P. (2023) Accord-Project, 2023. Available from: <https://doi.org/10.5281/zenodo.10210022>. Available from: <https://doi.org/10.5281/zenodo.10210022>.
- Hettiarachchi, H., Dridi, A., Gaber, M.M., Parsafard, P., Bocaneala, N. & Breitenfelder, K. et al. (2024) CODE-ACCORD: A Corpus of Building Regulatory Data for Rule Generation towards Automatic Compliance Checking.
- HuggingFace (2018) BERT base uncased, 2018. Available from: <https://huggingface.co/google-bert/bert-base-uncased>.
- Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. & McClosky, D. (2014) The Stanford CoreNLP Natural Language Processing Toolkit, 55–60. Available from: <https://doi.org/10.3115/v1/P14-5010>.
- Ramshaw, L. & Marcus, M. (1995) Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*. In *Third Workshop on Very Large Corpora*. Available from: <https://aclanthology.org/W95-0107.pdf>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. & Gomez, A.N. et al. (2017) Attention Is All You Need.
- Wang, X. & El-Gohary, N. (2023) Deep learning-based relation extraction and knowledge graph-based representation of construction safety requirements. *Automation in Construction*, 147, 104696. Available from: <https://doi.org/10.1016/j.autcon.2022.104696>.
- Xue, X., Zhang, J. & Chen, Y. (2024) Question-answering framework for building codes using fine-tuned and distilled pre-trained transformer models. *Automation in Construction*, 168, 105730. Available from: <https://doi.org/10.1016/j.autcon.2024.105730>.
- Yang, F. & Zhang, J. (2024) Prompt-based automation of building code information transformation for compliance checking. *Automation in Construction*, 168, 105817. Available from: <https://doi.org/10.1016/j.autcon.2024.105817>.
- Zhang, R. & El-Gohary, N. (2022) Natural language generation and deep learning for intelligent building codes. *Advanced Engineering Informatics*, 52, 101557. Available from: <https://doi.org/10.1016/j.aei.2022.101557>.

Disclaimer/Publisher's Note

The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and do not reflect the views of the *Architecture, Buildings, Construction and Cities (ABC2) Journal* and/or its editor(s). *DFBI Journal* and/or its editor(s) disclaim any responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.