



Deliverable 2.3

Generic framework for digitalisation of fire safety codes

ITEA4 22003 FireBIM

Deliverable type	Document
Deliverable reference number	ITEA 22003 D2.3
Work Package	WP2
Submission date	2026/09/28
Dissemination level	Public
Abstract	This deliverable presents a generic framework for the digitalisation of fire safety regulations. The framework converts heterogeneous regulatory documents into machine-interpretable regulation graphs and executable SHACL-based rules, while retaining human-readable representations for



	expert validation. Several rule interpretation methods are investigated, including RASE, NLP, LLM-based approaches, and a neuro-symbolic AI pipeline that combines ontology-based semantic grounding, formal logic, and SHACL. The framework is connected to RDF-based building information, geometric computation, and automated compliance checking. Demonstrators show the application of the resulting digital rules to building models. The framework also supports semantic rule retrieval through a GraphRAG-based natural language interface.
Keywords	Fire Safety, Building Codes, Automated Compliance Checking, Digital Building Codes, SHACL

Partner	Author	Role
Eindhoven University of Technology (TU/e) (NL)	Alex Donkers	Semantic web expert
Buildwise (BE)	Cedric Driesen	BIM expert
DTU (DK)	Martina Jakubowska	BIM expert
DTU (DK)	Kristoffer Negendahl	BIM expert
VN2R (PT)	Jéssica Reis	BIM expert
ISEP (PT)	Tomás Jorge	Fire Safety Expert
VN2R (PT)	Jéssica Reis	BIM expert
D-Studio (BE)	Stefan Boeykens	BIM expert /
Eindhoven University of Technology (TU/e) (NL)	Ekaterina Petrova	WP2 Lead / BIM expert
Technical University of Denmark (DTU) (DK)	Anne Dederichs	Project Coordinator / Fire Safety Expert



Executive Summary

Deliverable 2.3 presents a generic framework for translating fire safety regulations into machine-interpretable digital rules. The objective is to provide a common approach that can be applied to different regulatory documents and national contexts, while keeping the resulting rules understandable and traceable for domain experts. The framework therefore does not assume a single input format or a single method for interpreting regulations.

The proposed workflow starts with the conversion of national regulatory documents into a machine-readable regulation graph. Different preprocessing and interpretation methods are required because regulations differ in structure and format between countries. The FireBIM Regulation Ontology provides a common representation of regulatory documents and their structure. The resulting RDF graphs can also be transformed back into human-readable HTML representations, allowing experts to inspect the digitalised regulatory document.

The next step is the interpretation of regulatory statements and their conversion into executable rules. Several approaches were investigated. These include manual RASE tagging, NLP-based interpretation, LLM-generated decision trees, and a neuro-symbolic AI approach. The latter combines linguistic processing, semantic grounding using the FireBIM Building Ontology, formalisation in First-Order Logic, and transformation into SHACL shapes. Intermediate representations are retained, and SHACL shapes can be converted back into diagrams or text. This enables human experts to validate the correctness of the digital rule.

The framework also shortly addresses the building side of compliance checking. IFC building models can be converted into Linked Building Data graphs, which can subsequently be enriched with topological and geometric information. Geometric functions can then be executed on the building graph and their results used by SHACL validation. This enables rules involving spatial relationships, quantities, and other computed properties.

Four demonstrators were used to test the resulting workflow. These cover geometric fire compartment checks, compliance checking of heterogeneous building information, spatial containment and quantity constraints, and a combined GraphRAG and SHACL workflow. The demonstrators show that digital rules can be executed against RDF-based building information and that resulting violations can be reported in a structured way.

Finally, the regulation graph is enriched with thematic links to concepts from the FireBIM Building Ontology. This enables rule retrieval based on domain concepts rather than only the structure of the regulatory document. A GraphRAG implementation allows users to formulate natural-language questions, which are translated into SPARQL queries. The retrieved regulatory text is returned directly from the knowledge graph, while the generated query remains visible to the user.

The resulting framework provides the basis for the subsequent development of FireBIM software and machine-readable rulesets. It also establishes the connection between the terminology and ontologies developed earlier in WP2 and the implementation activities that follow.



Table of Contents

Executive Summary	3
Table of Contents	4
List of Acronyms	5
List of figures	7
List of tables	7
List of listings	7
1. Introduction.....	9
1.1. Objectives of Deliverable 2.3.....	10
1.2. Positioning within Work Package 2 (WP2).....	10
1.3. Proposed method	11
2. Background and state-of-the-art	12
2.1. Existing methods and approaches to compliance checking	12
2.1.1. Automated Compliance Checking (ACC)	12
2.1.2. BIM-based ACC	13
2.1.3. Rule-based approaches	13
2.1.4. AI-assisted ACC	14
2.2. Transforming regulations into checkable rules	15
2.2.1. Formal Representations	16
2.3. Rule-checking languages	16
2.4. Existing tools.....	17
3. Methodology	19
3.1. Digitalisation workflow and conceptual architecture.....	19
3.1.1. Stakeholder perspectives	19
3.1.2. Digitalization workflow	19
3.1.3. Validation	21
4. Results	23
4.1. Regulation graph	24
4.1.1. Preprocessing.....	24
4.1.2. Regulatory document interpretation methods.....	26
4.1.3. Semantic structuring of the documents using the FireBIM Regulation Ontology	27
4.1.4. Human-readable formats	29
4.2. Rule interpretation and digital rule generation	30
4.2.1. RASE	30
4.2.2. NLP-based interpretation pipeline	34



4.2.3.	Expert-driven diagram-to-SHACL workflow	35
4.2.4.	Semi-automatic diagram-to-SHACL workflow	35
4.2.5.	Neuro-symbolic AI method	37
4.3.	Building graph generation and geometric computations.....	41
4.3.1.	IFC to LBD conversion.....	41
4.3.2.	Procedures for integrating geometric functionalities.....	42
4.4.	Demonstrators.....	42
4.4.1.	Geometric compliance checking	42
4.4.2.	Geometric compliance checking of heterogeneous building information.....	44
4.4.3.	Spatial containment and quantity checks	45
4.4.4.	GraphRAG-Driven Rule Retrieval and Compliance Checking.....	46
4.5.	Rule exploration and retrieval.....	47
4.5.1.	Semantic enrichment and thematic metadata tagging	47
4.5.2.	GraphRAG architecture for rule retrieval.....	48
5.	Conclusion	50
	References.....	52

List of Acronyms

Acronym	Definition
---------	------------

ACC	Automated Compliance Checking
ACCORD	Automated Compliance Checks for Construction, Renovation, or Demolition Works
AECO	Architecture, Engineering, Construction and Operations
AI	Artificial Intelligence
API	Application Programming Interface
BCF	BIM Collaboration Format
BDD	Behaviour-Driven Development
BERA	Building Environment Rule and Analysis Language
BERT	Bidirectional Encoder Representations from Transformers
BIM	Building Information Modelling
BIMQL	BIM Query Language
BIMRL	BIM Rule Language
BMC	BIM Model Checking
CI	Continuous Integration
CSV	Comma-Separated Values
CSS	Cascading Style Sheets
DSML	Domain-Specific Modelling Language
EIR	Exchange Information Requirements
FBO	FireBIM Building Ontology
FOL	First-Order Logic

**Acronym Definition**

FRO	FireBIM Regulation Ontology
GAF	Generalised Adaptive Framework
GraphRAG	Graph Retrieval Augmented Generation
GUI	Graphical User Interface
HTML	HyperText Markup Language
IFC	Industry Foundation Classes
IFC2LBD	IFC to Linked Building Data
IDM	Information Delivery Manual
IDS	Information Delivery Specification
LBD	Linked Building Data
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MVD	Model View Definition
NLP	Natural Language Processing
OMG	Object Management Group
OPCTU	Object, Property, Comparison, Target, Unit
OWL	Web Ontology Language
PDF	Portable Document Format
RDF	Resource Description Framework
RDFS	RDF Schema
RASE	Requirement, Applicability, Selection, Exception
RIR	Regulatory Information Requirements
SDK	Software Development Kit
SHACL	Shapes Constraint Language
SMC	Solibri Model Checker
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SWRL	Semantic Web Rule Language
TTL	Turtle
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WP2	Work Package 2
XML	Extensible Markup Language



List of figures

Figure 1 Placement of D2.3 relative to other deliverables in Work Package 2	11
Figure 2 Global FireBIM framework dataflow	11
Figure 3 FireBIM digitalization workflow diagram.....	20
Figure 4 Seven validation activities mapped onto the digitalization workflow	22
Figure 5 Overview of the elements created by this framework	23
Figure 6 FireBIM Regulation Ontology.....	28
Figure 7 Instantiation of the FRO ontology using the Dutch construction law	28
Figure 8 Visual comparison of generated HTML (left) and original PDF (right).....	30
Figure 9 Interactive HTML representation of the Regulation graph in Portugal.....	30
Figure 10 A RASE-tagged Danish fire regulation.....	31
Figure 11 NLP-based pipeline to generate SHACL shapes from (Dutch fire safety) regulations.....	34
Figure 12 Dependency tree and noun chunks of a Dutch fire safety regulation.....	34
Figure 13 The workflow discussed in section 5.2	36
Figure 14 An example of a generated flowchart	37
Figure 15 A neuro-symbolic pipeline for generating SHACL shapes	38
Figure 16 A resulting SHACL shape	39
Figure 17 Flowchart representation of a fire safety regulation	40
Figure 18 An example of automatic translation of SHACL (left) to text (right).....	41
Figure 19 An overview of the IFC2LBD-Neo converter	42
Figure 20 Visualization of a pipe segment that goes through a compartment boundary wall.....	45
Figure 21 Method for LLM-driven graph enrichment	47
Figure 22 GraphRAG-based dynamic SPARQL generation	48
Figure 23 FireBIM Chatbot as a GraphRAG implementation	49
Figure 24 FireBIM Chatbot as a GraphRAG implementation, part 2.....	49

List of tables

Table 1 A comparison between preprocessing formats	24
Table 2 OPCTU table of the Danish fire regulation.....	31

List of listings

Listing 1 Snippet of the .txt serialization of the Dutch regulations	24
Listing 2 Snippet of the .xml serialization of the Dutch regulations	25
Listing 3 Snippet of the Portuguese regulations.....	25
Listing 4 Snippet of the Belgian regulations in .md	26
Listing 5 An RDF description of a chapter of the Dutch building code	29
Listing 6 SHACL shape representing the RASE-tagged Danish regulation.....	34
Listing 7 Resulting SHACL shape of the Dutch fire safety regulation	35
Listing 8 SHACL shape of a Dutch fire regulation	43
Listing 9 Validation report for a space that is deliberately not located in a compartment	44
Listing 10 SPARQL query returning all instances of pipe segments that go through compartment boundary walls	44
Listing 11 RDF snippet of the intersection between a compartment boundary wall and a pipe segment	44
Listing 12 Data graph that violates article 4.51(2)	45
Listing 13 Validation report generated by pySHACL.....	46



Listing 14 Validation report of article 4.50 member 1	46
Listing 15 Validation report of article 4.51 member 2	46
Listing 16 Generated stakeholder-specific solution to the validation report (stakeholder: architect, project phase: early design)	47



1. Introduction

Fire safety engineering is a critical theme within the Architecture, Engineering, Construction and Operations (AECO) industry, addressing preventive and protective strategies to mitigate fire hazards and other risks to property, people, and the environment (Khan et al., 2020; Kincelova, 2019; Sabbaghzadeh et al., 2022; Wang et al., 2023). Implementing effective fire safety measures reduces the likelihood of emergencies and minimizes their impact, such as by facilitating safe evacuation from buildings or construction sites (Sabbaghzadeh et al., 2022).

Fire safety engineering is crucial in building projects but often overlooked or addressed too late in the design process, leading to delays, higher costs, and health risks, including potential casualties (Khan et al., 2020; Sabbaghzadeh et al., 2022). Fire safety planning is typically separated from the design phase, despite its integration during this stage significantly improving cost-effectiveness and safety (Khan et al., 2020; Sabbaghzadeh et al., 2022). Additionally, the decentralized nature of fire safety engineering, fragmented fire codes, and varying regulations and interpretations across geographic levels and stakeholder groups hinder its seamless integration into design workflows (Kruiper et al., 2024).

Digitalization in the built environment, driven by Building Information Modeling (BIM) (Borrmann et al., 2018; Eastman et al., 2011), is improving collaboration, standardization, and fire safety by enabling efficient utilization of and computations with building information, thereby allowing more accurate design and automation of design modifications (Zhang & El-Gohary, 2022; Sabbaghzadeh et al., 2022). Automated Compliance Checking (ACC), powered by Artificial Intelligence (AI), Natural Language Processing (NLP), and Machine Learning (ML), is further advancing rule interpretation and automation in compliance checking processes (Zhang & El-Gohary, 2022; Vakaj et al., 2023). However, significant challenges persist in automating regulatory information extraction, rule representation, and aligning regulations with the design data (Zhang & El-Gohary, 2022; Bloch et al., 2023).

In this regard, the fire safety compliance checking field requires significant methodological advancements to improve the efficiency and accuracy of the building design and the corresponding checking processes. Moreover, the variability of human translation and interpretation of the regulatory texts undermine the accuracy of code (Amor & Dimyadi, 2021). Additionally, the natural language format of regulatory documents (often available in PDF) creates challenges for ACC (Wang et al., 2023; Zhang & El-Gohary, 2022). Important to note are the intricacies of the clausal regulatory document structure with multiple backward and forward references to internal and external document parts and sources (e.g., tables, figures, calculation methods, national and ISO standards, etc.), domain-specific characteristics, and ambiguities. Addressing these issues would enable the development of automated fire safety compliance systems, improve effectiveness and streamline fire safety engineering workflows (Sacks et al., 2020).

Several ongoing European research initiatives and projects aim to address the above challenges. The Automated Compliance Checks for Construction, Renovation, or Demolition Works (ACCORD)¹ project aims to achieve digital building permits and compliance verification by leveraging BIM and other data sources. ACCORD is developing a semantic framework for European digital building permitting processes, regulations, data, and tools. This framework facilitates rule formalization and integrates existing compliance tools as microservices, ensuring consistency, interoperability, and reliability across various EU regulatory contexts. More specifically, the ACCORD-NLP framework consists of several pre-trained/fine-tuned machine learning models to perform various information extraction tasks from regulatory text. In a similar effort, the Digital Environment for Management of Permits and Compliance in Building and Construction (DigiChecks)² project focuses on creating a digital framework to streamline (digital) permit validation and approval systems in construction projects based on a standardized permit ontology, a tool based on the Object Management Group (OMG) standards, allowing various actors to model their processes within DigiChecks, standardized building permit rules and integrating these into a Permit

¹ <https://accordproject.eu/>

² <https://digichecks.eu/>



Service Application Programming Interface (API). The Digital Compliance (D-COM)³ project and network were initiated to further drive the digitalization regulations, requirements and compliance checking systems in the built environment, as well as their adoption. D-COM shows the importance of engaging with key stakeholders who inform and drive regulatory policy development, thereby defining the required ACC capabilities. Among the important contributions is a roadmap to establish a working and operating model. The project emphasizes building a community and fostering collaboration among stakeholders to advance ACC.

Despite the significant progress made in recent research and the related initiatives, several research and development gaps remain. Among others, the persisting lack of harmonization of regulations and standards that significantly vary across countries, regions and municipalities still creates complexities that hinder the development and adoption of streamlined ACC solutions as the latter need to continuously adapt to different regulatory environments. As such, the development of methods to seamlessly translate, interpret and align diverse regulations into a common machine-readable format, as well as ontologies that can handle such (jurisdictional) variabilities need further attention. Furthermore, the ambiguous and context-dependent natural language format of the regulatory documents remains challenging to translate into machine-readable rules in an automated manner. In this regard, novel methods for handling, context, exceptions, multiple references and implicit rules, potentially driven by the developments in NLP, need further attention.

1.1. Objectives of Deliverable 2.3

The primary objective of Deliverable 2.3 is to establish a generic, scalable framework for translating natural language fire safety codes into executable Shapes Constraint Language (SHACL) shapes.

A secondary, equally critical objective is to move beyond black-box AI. Pure automation in a legal context is a liability. The framework must produce digital rules that are legally traceable and computationally transparent. We achieve this by designing a neuro-symbolic pipeline that restricts AI interpretation using strict semantic logic, as well as by implementing various human-in-the-loop guardrails.

1.2. Positioning within Work Package 2 (WP2)

Deliverable 2.3 acts as the logical bridge within Work Package 2. It connects the theoretical domain knowledge gathered at the beginning of the project with the functional software and rulesets required at the end. Figure 1 shows how D2.3 is placed in relationship with other deliverables in this work package.

- **Upstream Dependencies:** This framework relies on the fire safety definitions established in D2.1 (Terminology) and the formal Linked Building Data structures provided by D2.2 (The FireBIM Ontology). However, to enable future extension, these ontologies should be replaceable, such that the framework would also work for new local contexts or new knowledge domains.
- **Downstream Deliverables:** The neuro-symbolic logic and human-in-the-loop validation workflows defined in this document are a starting point for software development in D2.4, and could be used as a set of software requirements. Furthermore, to develop the methods in this deliverable, a variety of machine-readable rulesets were tested, and the result of this deliverable will thus determine how those machine-readable rulesets (D2.5) will be structured.

³ <https://www.dcom.org.uk/>
ITEA4 22003 FireBIM
D2.3

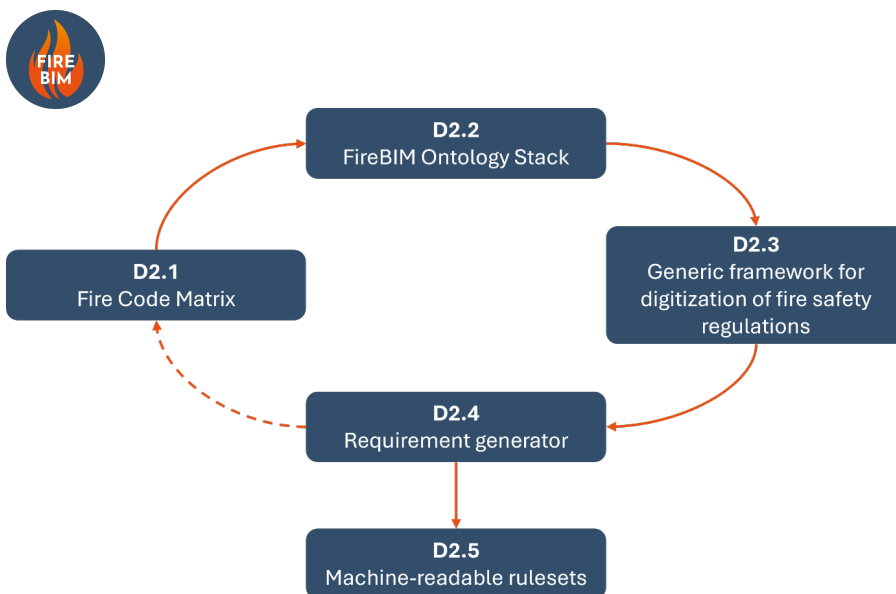


Figure 1 Placement of D2.3 relative to other deliverables in Work Package 2

1.3. Proposed method

The framework as defined in this document follows a simple architectural pattern, which is shown in Figure 2. It builds upon existing ideas on compliance checking (Amor and Dimyadi, 2021), where compliance checking is the interface between the domains of building (information) modeling and normative knowledge. In FireBIM, we ought to make both domains machine-interpretable, such that the act of compliance checking could be executed by a machine.

This deliverable shows results of a variety of methods that were tested to interpret rules, including manual tagging, RASE tagging, NLP methods, LLM methods, neuro-symbolic AI methods, a variety of semantic web methods, and other representations of symbolic logic. The hypothesis is that text-based regulations can be mapped to SHACL shapes, which is a machine-interpretable constraint language. Crucial in this interpretation is that the process is designed around human experts, so that expert knowledge is part of the interpretation of the text based rule and evaluation of the digital rule.

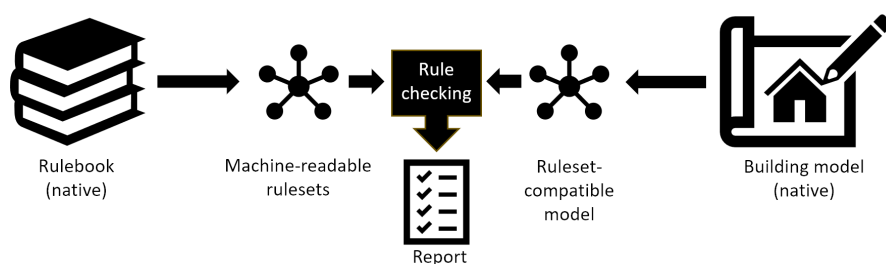


Figure 2 Global FireBIM framework dataflow



2. Background and state-of-the-art

2.1. Existing methods and approaches to compliance checking

Construction projects must comply with a series of requirements derived from building codes, regulations and design specifications. These requirements can be set by different entities and cover a variety of disciplines such as energy efficiency, safety, accessibility, among others. Manually verifying a project's compliance with all applicable requirements is a costly and time-consuming process prone to errors. In this context, Automated Compliance Checking (ACC) emerges.

2.1.1. Automated Compliance Checking (ACC)

ACC consists of a set of methods and tools that simultaneously process requirements in natural language and digital building models, analyzing/comparing both to detect instances of non-compliance (Amor and Dimyadi, 2021). Two decades ago, Eastman et al. (2009) proposed a general process for automatic compliance checking based on BIM models, structured in four phases: (1) BIM model preparation, where project data is standardized and adapted; (2) rule interpretation, which involves translating legal requirements into a language understandable by the system; (3) rule execution, where the model is evaluated and each item is compared with the defined criteria; and (4) communication of verification results through reports highlighting non-compliance and possible adjustments (Eastman et al., 2009; Zhang and El-Gohary, 2017). ACC is widely applied in various phases of the construction lifecycle, from design to maintenance. More specifically, in the design phase, Pauwels et al. (2011; 2016) proposed a method to assess and inspect the acoustic performance of buildings, in which the contents of relevant standards were computerized into a machine-readable format using RDF. In the construction phase, Zhong et al. (2018) designed an ontology for the quality inspection of construction projects, and Lu et al. (2015) used Ontology Web Language (OWL) to computerize safety standards in construction. In the maintenance phase, Zhong et al. (2018) focused on environmental monitoring and developed four ontologies to represent building information, sensors, and regulations, based on national standards and design requirements.

Additionally, some studies have begun to explore ACC in the context of fire safety regulations or related domains (Fitkau and Hartmann, 2024; Ismail et al., 2023; Chen and Jiang, 2024; Wang et al., 2023; Goh et al., 2024). For example, Wang et al. (2023), proposed a BIM and NLP-based ACC system to automatically check the spatial geometric relationships and property information of building models against fire codes, demonstrating high performance in precision and recall. Fitkau et al. (2024) developed a fire safety ontology (FiSa), which formalized the knowledge and facilitated the process of automatic compliance checking for structural fire safety requirements. However, these approaches are still relatively simple and have not yet been fully integrated into more comprehensive ACC frameworks. These early studies are important for laying the foundation, but further research and development are needed to address the complexity and nuances of fire safety compliance verification. More recent work combines domain ontologies and Semantic Web technologies with automated rule interpretation, enabling fire-safety regulations to be linked directly to machine-readable building information and executable compliance rules (Donkers and Petrova, 2024; Donkers and Petrova, 2026).

In general, regardless of its application, ACC's main goal is to streamline compliance analysis, minimize errors, and ensure uniform interpretation of regulations, promoting time savings, process standardization, and greater traceability in design decisions (Peng and Liu, 2023). On the other hand, the main challenges are related to the interpretation and formalization of complex regulations, integration with different formats and software, and the need to keep rules constantly updated.

Recent research increasingly positions ACC within the broader digitalisation of building permitting, rather than as an isolated model-checking task. Semantic Web technologies support the representation and linking of regulatory documents, building information, compliance rules, and permitting processes, providing a basis for interoperable and machine-processable permit workflows (Werbrouck et al., 2026). Recent ontology-based approaches similarly extend compliance checking beyond individual rules towards representing regulatory requirements, processes, and responsibilities (Hagedorn et al., 2025).



An example of digitization and compliance checking is the platform “UpCodes Web,” developed by UpCodes, a startup that combines data and software development to create tools for architects (Cheng, 2020). UpCodes Web is a search engine enhanced with machine learning and analytics, designed to help users find the applicable building codes, though it currently covers only certain U.S. jurisdictions. In practical terms, UpCodes serves as an online database aimed at improving the process of understanding construction codes and design interpretations. The platform also includes an AI-powered code research assistant that helps users navigate codes and find answers to specific questions. In addition to these features, UpCodes offers a Code Calculator, divided into four modules: “1 - Heights and Areas,” “2 - Egress,” “3 - Fire Ratings,” and “4 - Plumbing Fixtures.” After entering project details, users can input building information either manually or by importing a pre-populated CSV file exported from Revit, in line with the platform’s guidelines. Once the information is entered, the Code Calculator generates a report that includes module-specific analysis, footnotes, non-compliant conditions, and step-by-step calculations.

2.1.2. BIM-based ACC

The preparation of the BIM model is crucial for the success of ACC, enabling more rigorous and efficient automatic compliance checks. BIM provides a comprehensive data foundation, including geometry, materials, and properties, making analysis more reliable and facilitating the early detection of non-compliances and reducing rework. As demonstrated by Eastman et al. (2011), the use of BIM significantly improves coordination and information management in complex projects. Additionally, a study conducted by Sacks et al. (2010) highlighted that the application of BIM reduces design conflicts by up to 45%, saving time and resources.

For BIM to be effective in ACC, it is essential to ensure that the model is properly standardized, as well as using interoperable formats such as IFC. It is also crucial to simplify the model by removing unnecessary elements and associating relevant semantic information for better interpretation. The success of this process fundamentally relies on rigorous modelling practices, interoperability between platforms, and constant updating of regulatory knowledge bases.

In addition, Jos (2023) categorizes BIM model checking (BMC) methods into three main types: Clash Detection, Model Content Checking, and Code Checking. Within Code Checking, there are three subcategories: Geometry-based Checking, Location-based Checking, and Functionality-based Checking. Geometry-based checking ensures components are accurately sized and shaped, location-based checking verifies correct placement and spatial relationships, and functionality-based checking validates the operational performance of the building system.

Furthermore, as addressed by Henttinen (2024), buildingSMART introduced the Regulatory Information Requirements (RIR) initiative to ensure that the buildingSMART IFC standard addresses common building regulation needs. The goal is to improve the pre-checking of applications and restructuring compliance checking processes, while still allowing jurisdictions to add local information requirements and implement their own checklists and rules. Regulatory Information Requirements include a group of recommendations with a focus on general identification and location, life safety considerations (including fire safety and accessibility), and performance standards (with an emphasis on environmental sustainability and efficiency). The recommendations are available in two key formats: first, as a Use Case IDM with an accompanying IDS file that can form the basis for localisation. Second, they are available as properties and user-defined types within the buildingSMART Data Dictionary (bsDD) under the context of "buildingSMART Regulatory Information Requirements", offering a standardized framework for managing regulatory information in Building Information Modeling (BIM) processes.

2.1.3. Rule-based approaches

Depending on the complexity and category of the rules (mathematical, conditional, relational, etc.), various methodologies can be used to check if a project complies with regulatory requirements. Each method has its advantages and disadvantages, and the choice depends on the specific needs of the project and the available resources. Approaches range from simple methods, such as manual or tool-assisted analysis, which rely on practical examples to identify obvious flaws in projects, to more advanced methods, such as machine learning, which uses large datasets to identify complex patterns and predict potential non-compliances. Other strategies include constraint-based approaches, where clear limits are defined for verification, and model-based



methodologies, which use ontologies and metadata to describe semantic relationships between project elements (Dimyadi et al., 2014).

Among these alternatives, rule-based methodologies stand out, which adopt "if-then" instructions or declarative rules to evaluate each element of the model. Formalizing these rules requires creating logical clauses that reflect regulatory requirements, often using inference engines (e.g., SWRL), ontologies (e.g., Protégé), or specific languages for ACC. Several studies have proposed rule-based approaches to automatically verify specific aspects. Lee et al. (2018, 2021) created a computer-readable rule base and an automatic prototype, using the *IfcDoc* tool, to verify compliance with the *Model View Definition* (MVD). To check MVD consistency, Zhang et al. (2015) developed a prototype based on rules in XML format. In another study, Fahad et al. (2018) proposed a method that converts IFC models to the *Resource Description Framework* (RDF) format. From this conversion, they created specific vocabularies to represent the items to be verified and established a rule base to identify and correct semantic errors. Luttun et al. (2020) proposed a method based on Boolean operations to identify geometric errors in IFC models. The approach calculates the geometric relationships between elements and compares them with the correct expected relationships.

All these studies demonstrate that, although rule-based approaches offer a high level of accuracy and control in the verification process, they also require significant adaptation effort due to the ambiguity of natural language, the complex structures of documents, and the incompatibility between BIM model languages and regulations.

2.1.4. AI-assisted ACC

To mitigate the ambiguity of natural language, the complex structures of documents, and the incompatibility between BIM model languages and regulations, *Natural Language Processing* (NLP) and *Artificial Intelligence* (AI) have taken on an increasingly important role in the context of ACC, significantly enhancing the automation and efficiency of processes. NLP, by combining linguistics and computer science, allows computational systems to automatically process, analyze, and interpret natural language texts, while AI enables the development of systems that can interpret external data, learn from it, and perform tasks that typically require human intelligence. In NLP, advanced techniques are transforming how regulations are processed (Ismail et al., 2023; Guo et al., 2021). Zhang et al. (2022) highlight that NLP enables accurate semantic extraction, reducing the manual effort required to convert standards into computational rules. For example, the use of pre-trained models like *BERT* provides remarkable performance in analyzing regulatory clauses by accurately identifying relationships between textual entities in complex contexts. However, it is important to note that these models come with high computational costs, which may limit their application in broader scenarios.

AI, in turn, complements NLP by eliminating the need for manual programming to identify patterns in unstructured data. Models such as TextCNN are highly efficient for analyzing simple texts, while LSTM excels in more complex contexts due to its ability to handle long-range sequential dependencies (Zhang and El-Gohary, 2022). Hybrid architectures, such as fusion models, integrate the advantages of various approaches to achieve greater precision and accuracy, though they require more training time due to the complexity of their structures (Chen et al., 2024).

Recent research has increasingly shifted from conventional NLP and supervised classification towards transformer-based models and Large Language Models (LLMs) for interpreting and formalising building regulations. These approaches have been applied to semantic tagging, rule interpretation, BIM-based compliance checking, and direct generation of executable checks. Recent studies include fine-tuned LLMs combined with retrieval mechanisms (Shi et al., 2025), LLM-based BIM compliance checking (Madireddy et al., 2025), and end-to-end LLM-centric compliance-checking frameworks (Iversen and Huang, 2026).

Despite their significant potential to streamline and improve analysis, important challenges persist. The availability of labelled data, essential for training machine learning algorithms, is often limited, making it difficult to develop more robust models. Furthermore, the intrinsic complexity of regulations and the need for rigorous contextual interpretation make human supervision and validation indispensable—key elements to ensure the reliability, transparency, and legitimacy of decisions made by these advanced systems.



2.2. Transforming regulations into checkable rules

Kommenterede [EPI]: @Kristoffer, Martina

Regulatory texts are filled with ambiguities, which significantly complicate the interpretation and translation into computable rules. Zhang et al. (2023) identified four types of ambiguity in building regulations. These ambiguities can either be intentional, for example for the purposes of promoting design flexibility, or maintaining validity against technological changes, or unintentional. The first class of unintentional ambiguities stems from the usage of natural language, and covers issues such as vagueness, incompleteness, and lexical ambiguity, that is, a word or a phrase having multiple meanings. The two other kinds of ambiguity identified were tacit knowledge and ACC-specific issues. Tacit knowledge refers to the implicit knowledge assumed to be shared between regulators and the designers and engineers using the regulations. Some of this knowledge cannot be easily formalised or written down, other parts are potentially explicable. According to Zhang et al. (2023) regulators assume a certain level of domain knowledge and try only to specify the non-obvious parts, which in result requires that these rules are checked manually by experts using their tacit knowledge. Lastly, there are ambiguities specific to the automatic compliance checking domain. This category mostly covers over ambiguities in mapping objects and relationships between the rules and building object models. One-to-one direct mappings are not always possible, there may be one-to-many relationship or there may not be any direct mapping between a regulatory concept and the model.

Nawari (2019) pointed out that most of the over 400 relevant research studies in the ACC domain, propose approaches that do not support dealing with ambiguous information and can be categorised as either black box or gray box. In this context, black box refers to methods being dependant on proprietary schemas, rule checking engines or software, thus resulting in a system that is opaque, difficult and expensive to maintain or modify. One of the key approaches to developing a white-box interpretation of regulatory texts is leveraging logical reasoning methods, primarily based on predicate logic, such as First-Order Logic (FOL). These methods are inherently deterministic, making them a good choice for ensuring transparency and trustworthiness in the system.

Focusing both on the interpretation of ambiguous information, supporting open data standards like IFC and utilising FOL, Nawari proposed a Generalised Adaptive Framework (GAF) for automating or semi-automating the code compliance verification process (Nawari, 2019). Lee et al. (2016) presented a predicate logic-based methodology for converting the Korean Building Act's textual regulations into a computer-executable format to facilitate automated permit evaluations. A central part of the KBimLogic module is creating a mapping between the original law text and IFC model objects. Their approach ensures that IFC data can be directly queried and evaluated using logical rules, reducing ambiguity in regulatory interpretation. Yang et al. (2022) proposed a new Information Representation and Reasoning (InfoRR) schema leveraging the newest advancement in logic programming for building code compliance. The proposed schema splits the information into two parts; logic facts and logic rules and compares the applicability of two logic languages; Picat and B-prolog, to ACC. Zhang and El-Gohary (2017) proposed combining automated NLP techniques for processing the regulatory text, EXPRESS data processing for accessing information from IFC models and FOL-based semantic reasoning, using a B-prolog reasoner, for checking the compliance of logic facts with the logic rules.

A noteworthy approach to capturing normative rules in regulatory texts is the semantic mark-up methodology, RASE (Hjelseth, 2012; Hjelseth and Nisbet, 2010). It can be used to capture the logical structure of regulations in natural language by tagging parts of a sentence as a Requirement, Applicability, Selection or Exception. This tagging process should be performed by domain experts with a deep understanding of the regulatory text and its inherent assumptions. The methodology was designed to be both practical and intuitive for non-technical experts, thus ensuring that critical domain-specific knowledge is preserved. Each RASE statement corresponds to one of four logical operators: AND, OR, NOTAND, or NOTOR. As a result, these statements can be automatically translated into predicate logic and computable rules expressed in procedural programming. This process provides a high degree of confidence in the system, as it minimises the risk of significant information loss through the transformation steps and allows experts to examine the exact logic underlying the computable rules in detail.



Alves de Mendonça (2024) integrated the RASE method with exchange information requirements (EIRs) expressed in the new, XML-based buildingSMART standard Information Delivery Specification (IDS) for an easier linking between regulatory objects and modelling requirements (de Mendonça and Ferreira, 2024). Beach et al. (2015) extended the RASE method to a comprehensive regulatory compliance system supporting mapping to a Web Ontology Language (OWL) version of the IFC schema and conversion to Semantic Web Rule Language (SWRL) rules (Beach et al., 2015). Other experiments explored integrating RASE with SPARQL queries and Natural Language Processing (NLP)-based information extraction with varying results (Nisbet et al., 2023; Zhang and El-Gohary, 2016).

The availability of suitable datasets is another limitation for automated rule interpretation. CODE-ACCORD, for example, provides a corpus of 862 self-contained regulatory sentences from English and Finnish building regulations specifically intended to support machine learning and rule-generation research (Tan et al., 2025).

2.2.1. Formal Representations

To enable encoding regulatory rules into computable formats, the rules must first be formalised, so the logical semantics can be extracted (Dimyadi et al., 2014). However, Garrett et al. (2014) pointed out the importance of these formalised representations being non-proprietary and user-understandable, otherwise “many of these computable versions of regulations might go unverified, assumptions will not be understood or managed, and ambiguities may lead to misinterpretations”.

Fuchs et al. (2024) investigated the feasibility of developing formal representations of building regulation at the same time as the natural language versions through qualitative interviews with experts involved in developing building regulations from five countries. They tried to establish how different formal representations are perceived and how they could be incorporated into the regulation drafting process. They presented two different representations for the interviewees; one code-like representation and one tabular semantic. The interviewees preferences of them varied based on their experience. It was pointed out that the code-like representation won't easily be understandable for lawyers, but in turn was viewed as more useful in that it would force the regulators to contemplate information requirements and suitability for ACC. Consequently, the need for interdisciplinarity in preparing building regulations was emphasised by the authors and interviewees alike. Specifically, three separate, necessary roles were identified: politicians and domain experts, legal experts and IT experts.

Zhang (2017) performed a quantitative study with 93 respondents with various backgrounds, surveying the perception of three different types of representations, namely text, logic-based and tree-based. They found that the tree-based representation improved understandability and reading speed compared to the others.

Nisbet et al. (2022) explored six different knowledge representations of RASE marked-up text. These included visual decoration of text in HTML using a CSS stylesheet, a checkable list, a code-like Datalex presentation, a logic gate presentation, a visual programming-based presentation and one relating RASE to formal knowledge through 'concept graphs'.

Recent reviews confirm a transition from predominantly rule-based compliance checking towards approaches combining symbolic representations, ontologies, machine learning, and LLMs. At the same time, interoperability, scalability, explainability, and reliable interpretation across jurisdictions remain unresolved challenges (Lee et al., 2026; Sabri et al., 2026).

2.3. Rule-checking languages

ACC is closely related to rule-checking in the legal domain and general-purpose rule-checking. Because this research area is extensive, a number of formal human-and-machine readable syntaxes for rule representation have been proposed throughout past decades. Notably, LegalRuleML is an extension to the popular RuleML markup language for rule formalisation, which includes features specific to legal norms, guidelines, policies and reasoning. LegalRuleML has also been successfully implemented in the AEC domain (Dimyadi et al., 2017).

Rule-languages in general often mix the rule representation and logic needed to execute them into one, sometimes allowing for multiple alternative visual representations on top of the source of truth. That is the case



with Semantic Web Rule Language (SWRL). SWRL is an integration of RuleML with the Web Ontology Language (OWL) and can express both the rules and their logic.

A number of proposals are domain-specific modelling languages (DSMLs) tailored to the ACC and BIM domains, specifically focusing on the IFC object model. These include the Building Environment Rule and Analysis Language (BERA) providing model-checking capabilities without the need for precise knowledge of general-purpose programming. BERA is however heavily inspired by Java, which may still pose a challenge for non-programmers (Lee, 2017). BIM Rule Language (BIMRL) is another proposed DSML which may have the same issue, as it heavily draws influence from SQL (Solihin, 2015). The above-mentioned research into Korea's predicate logic-based regulation checking introduced KBimLogic, a tool for assisting users in natural language parsing and information extraction and KBimCode, a computer-executable code language for the rules. OpenBIM Rule Language (OpenBIMRL) is an XML-based open format and method for defining and validating functional requirements to IFC models with a user-friendly visual programming representation of complex rules (Stepien et al., 2023). A related initiative is the open BIM query language (BIMQL) capable of on-the-fly compositions of queries of the IFC model. It's implemented on top of the bimserver.org framework but is generic enough to be adapted to other IFC-based implementation tools (Mazairac and Beetz, 2013).

The IfcGherkin syntax provides a framework for validating normative rules in IFC. It was introduced by Moulton and Krijnen (2020), who presented an approach that applies behaviour-driven development (BDD) and continuous integration (CI) principles from software development to compliance checking of IFC models. In it, building requirements were defined in the Gherkin syntax, an ordinary language parser central to BDD and the Cucumber tool widely used in unit testing of software. The strength of (Ifc)Gherkin is the ability to define rules in a non-technical and human readable way, and linking them with programmatic definitions, thus making them available for all stakeholders. Today, IfcGherkin is central to the buildingSmart Validation Service.

More recent approaches increasingly combine Linked Building Data, ontologies, and SHACL to represent regulatory requirements as interoperable and executable constraints. For example, recent work demonstrates the use of standardized semantic templates to link building-code requirements to LBD models and validate them using SHACL (Zentgraf et al., 2026).

2.4. Existing tools

Currently, there are several dedicated software tools (CORENET BIM e-Submission, BIM Assure, Jotne Express Data Manager (EDM), Solibri Model Checker (SMC), and Daima) that enable the complex task of compliance checking to be performed automatically and efficiently. All these tools were developed using hard-coding techniques. The CORENET project in Singapore is recognized as the first successful initiative in this field (Khemlani, 2005). Initially conceived as a web-based service for verifying 2D drawings (CORENET BP-Expert), the project evolved into 3D model checking (CORENET e-Plan Check), now known as CORENET BIM e-Submission (BCA, 2021). Similarly, in Australia, the DesignCheck (Ding et al., 2006) was developed, also intended for 2D drawing verification. Although there were plans to expand its capabilities to 3D models, the project was eventually discontinued (Dimyadi and Amor, 2013).

Over the years, various software tools have emerged specifically designed to analyze and verify building models on BIM platforms. Among these, SMC stands out, an independent application that offers around 50 predefined rules (but configurable into endless variants) covering aspects such as evacuation, space management, and accessibility (Nawari, 2018; Smith, 1991). This software allows users to select rule sets, adjust parameters, and with a simple click of the "CHECK" button, perform automated checks. SMC was tested in the HITOS project in Norway (Eastman et al., 2009) and is currently widely used for BIM model validation in the United States of America (GSA, 2020). Other notable tools include BIM Assure, launched in 2016 (Invicara, 2020), and Daima, launched in 2019 (Daima, 2020), both developed as plugins for Revit. BIM Assure stands out by offering support for a cloud-based environment.

In the BRISE-Vienna project (Fisher et al., 2023), Solibri Office was integrated into an openBIM framework to facilitate model quality assurance and ensure adherence to building codes and project specifications. However,



the predefined rule sets within Solibri did not fully accommodate the specific requirements of Vienna's building code. As a result, custom rules were developed through Solibri's Application Programming Interface (API), allowing for a more accurate alignment with the local regulatory standards.

However, a significant limitation of existing tools is the restriction on flexibility for users to create new rules. In CORENET BIM e-Submission, Singapore's building regulations are represented using an object-oriented approach, implemented on the standalone FORNAX platform (Eastman et al., 2009; Khemlani, 2023). FORNAX is a C++ object library where rules are directly coded into the system and integrated with IFC objects. In DesignCheck, Jotne EDM provided a rule engine on the backend to represent and manage the rules, but modifications could only be made by programmers (Eastman et al., 2009; Preidel and Borrmann, 2018; Shih et al., 2013). SMC faces similar challenges: users can only adjust parameters of predefined rules (Nawari, 2018; Lee et al., 2015), either using the Java SDK to develop them as plug-ins or, alternatively, requesting their addition as a Solibri Extension in the Solibri Solution Center, which may require a paid subscription (Nawari, 2018; Preidel and Borrmann, 2018; Greenwood et al., 2010). In BIM Assure, the creation of new rules is limited to non-geometric parameters (Kincelova et al., 2019), and in Daima, this functionality is also exclusive to programmers (Daima, 2020).

Although the early applications demonstrated the reliability of hard-coding techniques, maintaining and expanding the rules proved to be complex tasks. As a result, transparency and flexibility in accessing rule databases have become valued characteristics in more recent automated systems. Previous research suggests that adopting a logical approach to rule representation would make systems more accessible to users.



3. Methodology

3.1. Digitalisation workflow and conceptual architecture

Automated compliance checking can only happen when regulations appear in a digitalized format. To reach this state, FireBIM proposes a pipeline to digitalize fire safety regulations. The following subsections describe how we developed the framework that is presented in section 4.

3.1.1. Stakeholder perspectives

Two workshops were organized, in which all international FireBIM participants were invited. These workshops were held in Lund, Sweden in June 2024 and in Eindhoven, the Netherlands in November 2024. During those workshops, participants from different domains came together, including but not limited to:

1. Fire safety engineers
2. BIM engineers and consultants
3. Software developers
4. Linked data engineers
5. Business/commercial roles

coming from both industry and research institutions. During those workshops, focus sessions with domain experts were alternated with sessions in which cross-pollination between the domains took place. This was particularly helpful in improving understanding of the challenges and wishes between the different domains. An outcome of these workshops is a general understanding of those challenges and wishes, which materialized into a set of 'starting points' for developing the regulation conversion pipeline.

- An expert should be able to validate all steps in the regulation conversion process.
- Metadata of the digitalized rule should indicate how the regulation conversion process was validated.
- Digitalized rules should be both machine- and human-understandable.
- Digitalized rules should be as software agnostic as possible.
- Digital rules and terminology should be abstract enough to be usable across European borders and simultaneously be detailed enough to be valuable in national or even local automated compliance checking.
- Digital rules and terminology must, where possible, be compatible with existing standards from various domains, such as ISO 13943:2023 and ISO 19650.

3.1.2. Digitalization workflow

An abstract overview of the digitalization workflow proposed in this project is found in Figure 3. The workflow is separated into four blocks that can be individually executed and validated. As the input for this workflow (regulations in any form, typically a combination of PDF and HTML files and accompanying images and tables) is different for each country, and will likely even change within different local contexts, the exact execution of each of those blocks will not be fully reproducible for other countries.

Kommenterede [TJ2]: Digitalisation or Digitization? the terms should be standardised

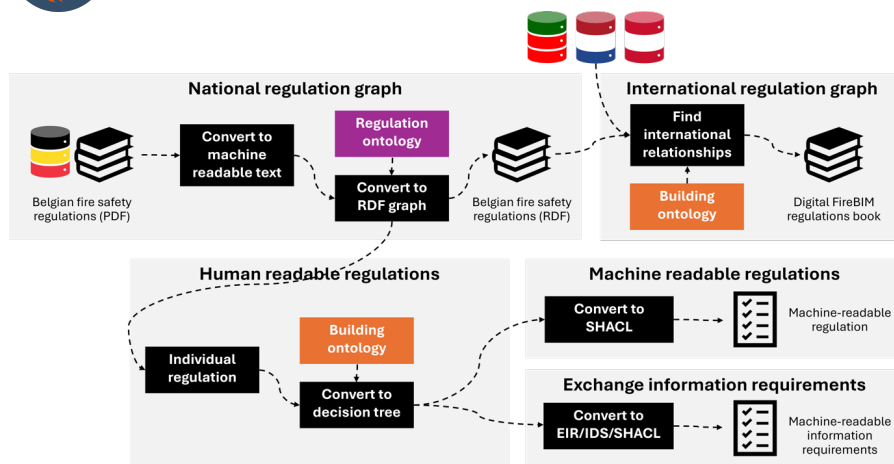


Figure 3 FireBIM digitalization workflow diagram

3.1.2.1. National regulation graph

The first block ('National regulation graph') aims to convert the national or local regulatory documents to a machine-understandable structure. The input for this block consists of a variety of regulatory documents in multiple formats, such as PDF documents, tables and figures, HTML pages, and so on. These are first converted to a format that is interpretable by a machine, after which they are converted to an RDF graph that follows the FireBIM Regulation ontology. This Regulation ontology aims to structure the regulatory documents, and contains concepts such as 'chapter' and 'article'. The content of regulations, covering concepts related to BIM ('building', 'escape route') and fire safety ('fire extinguisher', 'fire compartment'), and thus the actual understanding of those regulations, is outside of the scope of this step. It merely gives a common structure to the individual articles and their interrelationships.

3.1.2.2. Human-readable regulations

The second block ('Human readable regulations') aims to convert the content of those individual articles to a structured, human-readable format. Based on the analysis of building regulations, and the typical step-wise approach that they follow, such human-readable format is expected to be a diagram or flowchart. These flowcharts are enriched with concepts from a second ontology, the FireBIM Building ontology. This ontology, which is more of a modular stack of ontologies, covers concepts related to buildings and fire safety in buildings, and can easily be extended by other modular ontology snippets to cover other domains in the future. The ontology enables semantically tagging the content of the individual regulations and thus enables generating a common understanding of concepts in those sentences (such as 'fire compartment' or 'area of a zone'). This is important, because in the current practice, definitions of such concepts change per country or even per person, and how such definition materialized in practice is influenced by the expert's individual understanding of the concept. Automated compliance checking, however, has difficulties with such fuzzy definitions and benefits from a more rigid set of common concepts. This second block exports flowcharts with semantics. An important reason to include this step in the workflow – instead of directly converting to machine-understandable regulations – is so that experts can validate the semantic tagging procedure. Next to this, such flowcharts might also increase the understanding of regulations and enable a deeper comparison of regulations across countries.

3.1.2.3. Machine-readable regulations

The third block ('Machine readable regulations') converts the flowcharts to SHACL (Shapes Constraint Language) shapes. Those SHACL shapes can later be used to validate whether a building complies with the regulations. SHACL



is the constraint language in the semantic web stack and allows checking the shape and content of RDF graphs against a predefined 'shape'. Those shapes can contain constraints that data must conform to, including types, properties, values, metadata requirements, and mathematical operators. Such constraint checking cannot be performed by other languages in the semantic web (such as RDFS or OWL) as they operate in the open world assumption and can therefore only suggest new information (through inferencing) but lack the capability of restricting data.

3.1.2.4. Exchange information requirements

The fourth block ('Exchange information requirements') is closely related to the third block and creates a similar set of rules that can be used to validate the data. However, the intention of this fourth block is to check whether the building information is compliant with a predefined set of information requirements. Such exchange information requirements (EIRs) are typically checked prior to the ACC procedure. Such EIR-procedures are standardized in ISO 19650 and can be executed in multiple ways. buildingSMART developed the *Information Delivery Specification* (IDS) as a machine-readable format to perform EIR, and it is recognized as the standard procedure to perform EIR validation on IFC files. EIRs can also be serialized as SHACL shapes, in order to validate the content of RDF graphs. As this project intends to enable fire safety checking in various phases of the design lifecycle, and information comes in different shapes and formats during the different lifecycle phases, the digitalization workflow should be able to produce EIRs in different formats.

3.1.2.5. International regulation graph

The final fifth block ('International regulation graph') of this workflow aims to semantically tag the structured regulations (resulting from the first block) with concepts from the building ontology and afterwards bringing regulation graphs from multiple countries together. This enables querying regulations based on a concept (such as 'fire compartments') and finding similar regulations from other countries. Not only do we intend to enable automated compliance checking; semantically enriching regulations also opens the possibility to explore other regulations from other countries, learn from other perspectives, find smart solutions to difficult fire safety engineering tasks abroad, and share knowledge internationally. Those 'softer tasks' are enabled by this international regulation graph, and lead to the Digital FireBIM regulations book, which will be made available in both a machine-understandable (RDF graph) and human-understandable (GUI) format.

3.1.3. Validation

One of the most important facets of this procedure is that end-users should be able to trust the digital version of a regulation, in order to be able to use it. FireBIM therefore proposes a validation procedure that enables checking whether machines or humans correctly converted data to another format. Figure 4 shows a visual representation of the validation activities, mapped onto the digitalization framework. We distinguish validation of the trustworthiness of the digital rules and validation of the rules against a BIM model (which is the actual compliance checking). The figure below just discusses the trustworthiness of the digital rules. It does also not validate the content of the ontologies used in the digitalization procedure, but only the application of them in converting data. The content of the ontologies is validated in the design process of those ontologies, which is outside the scope of this deliverable.

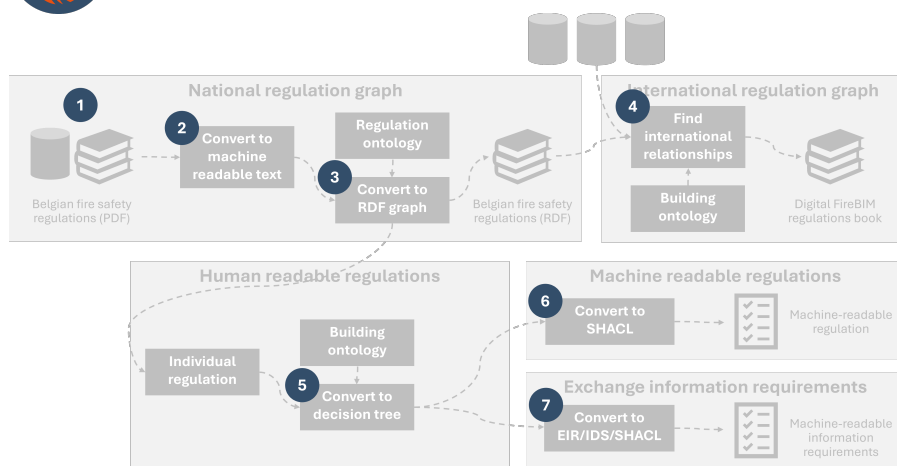


Figure 4 Seven validation activities mapped onto the digitalization workflow

Step 1: An expert should check whether the correct regulations are used as a starting point of the digitalization procedure. This requires a metadata check: is this the correct document? Is this the correct version of the document? Is the issue date of the regulations correct? Do external figures and tables match the referenced figures and tables in the text, and are these also up to date?

Step 2: An expert should check whether the machine-readable text (such as Markdown) correctly corresponds to the original regulations. Is all the text converted? Is there a structure in the text, and is this structure correctly converted to the Markdown document? Was the text translated, and if yes, is this done correctly?

Step 3: An expert should check the conversion of the Markdown text to an RDF representation. This can for example be done by performing a visual analysis on snippets of the graph. Does the graph represent the same structure as the document? Is there a clear hierarchy between chapters, subchapters, articles, and so forth? Are there irregularities, such as separate floating parts of the graph? This could also include a technical check, using software tools, including an RDF syntax checker.

Step 4: Since the national graphs can be interconnected by drawing international relationships between parts of the digital regulations, international consortia of fire safety experts should be able to validate whether those links are correct. If two regulations have a specific content-related tag, do the experts think that this tag should exist? Are two articles really related to the same concept? Is the definition of concepts across borders similar?

Step 5: An expert should be able to check whether the conversion to the decision tree (or flowchart, such as the Mermaid diagrams) is performed correctly, and whether those flowcharts are correctly modeled. This can be done by comparing the visual diagrams with the original regulatory text. Next to the correct shape of the diagram, this step should also validate whether the semantic tagging of concepts inside this diagram was performed correctly. Are the correct terms from the FireBIM ontology used in the tagging procedure? This latter step might require the help of a BIM expert.

Step 6: A fire safety expert should be able to validate whether the SHACL shape representing the decision tree is correctly modeled. This step requires expert knowledge outside the domain of the fire safety expert and likely requires help from a semantic web expert. Various methods to check SHACL shapes could be part of this procedure, each adding to the trustworthiness of the conversion. Those methods include validating the shape's syntax using an RDF checker, checking whether the SHACL shape correctly reused the semantics from the diagram, whether the SHACL shape is correctly linked to the national regulation graph, and whether the metadata inside



this SHACL shape is correct. A demo use case could help to validate the shape, by deliberately violating a rule using a demo BIM model and validating whether the SHACL shape catches this violation. This step is especially important as it is the first time that the natural language (including all its inexplicitness and vagueness) is converted to a very structured constraint language, and it requires thorough understanding of this language to understand whether the natural language is converted correctly, and what the implications of this conversion will be in rule checking procedures. If a rule is purposefully vague, for example when an expert validation based on a local context is required, then its SHACL representation should also allow some of this vagueness. Finally, we propose a deterministic SHACL-to-text and SHACL-to-diagram pipeline, such that fire safety experts could ‘read’ or ‘see’ how a SHACL validator executes the digital regulation.

Step 7: Similarly, a fire safety expert, with the help of a semantic web or BIM expert, should be able to validate the Exchange Information Requirements. This is a more delicate procedure, as the EIR can come in multiple formats, and depending on the chosen format, a different validation procedure takes place.

Future work could also test whether an LLM could be used to validate regulations, albeit for academic reasons, and compare the performance of the LLM with the performance of the fire safety expert. It is currently unclear whether LLMs could be used to assist humans in the validation procedure, and what the performance of the LLM is compared to the fire safety expert (who will likely also not be 100% accurate due to human errors).

3.1.3.1. Transparency on the performance evaluation

In line with performance certification approaches such as BREEAM or LEED, there is the intention in the FireBIM project to enable making the evaluation of the digital rule explicit in the graph. To express this, a newly created FireBIM Regulation Quality Ontology is created. The instantiation of this ontology will give an explicit score on the performance evaluation of a specific digital rule, and can be used as an indicator of how the rule can be trusted: a higher score means more reliable rules.

4. Results

The FireBIM framework translates building regulations into a machine-interpretable format through several steps. The framework in its essence has four proposed results (Figure 5):

1. A regulation graph, that covers the graph-based regulatory document,
2. A rules graph, containing the machine-interpretable rules (and the EIR, potentially),
3. A linked building data graph, either real or virtual, containing RDF-based building information models
4. A compliance report, which is a result of a compliance check.

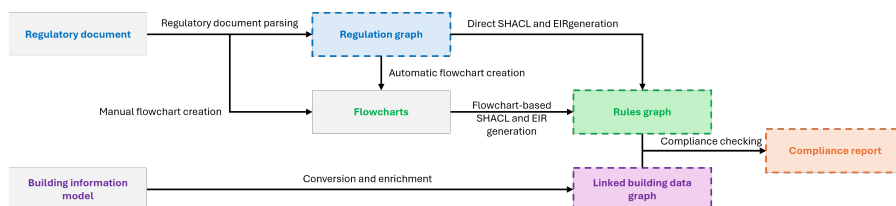


Figure 5 Overview of the elements created by this framework

The details of the framework are presented following these four elements. Section 4.1 covers the creation of the regulation graph, Section 4.2 covers the creation of the rules graph, Section 4.3 covers the creation of the linked building data graph, and Section 4.4 covers the compliance reporting and demonstrators. Finally, Section 4.5 covers rule exploration and retrieval methods, and Section 4.6 covers rule validation.



4.1. Regulation graph

4.1.1. Preprocessing

Because regulatory documents are published in a highly heterogeneous mix of formats across the various participating countries in FireBIM, applying a single pre-processing method is inefficient. The primary objective of the pre-processing stage is to convert these unstructured or semi-structured regulatory documents into editable text formats (such as .txt, .docx, or .md) that can then be used in the rule interpretation phase. Various formats were considered, and they are summarized in Table 1.

Table 1 A comparison between preprocessing formats

Format	Structure	Human-readable	Machine-friendly
.txt	Plain text, no formatting	Yes	Easy to parse
.rtf	Supports basic formatting	Yes	Needs a library
.md	Supports lightweight markup	Yes	Easy to parse
.xml	Hierarchical structure	No	Easy to parse
.json	Key-value hierarchical structure	No	Easy to parse
.docx	Complex structure OpenXML	Yes, but only in dedicated software tools such as MS Word	Needs a library

4.1.1.1. Dutch regulations

The Dutch regulations are available in structured .html and .xml formats. They therefore don't necessarily need preprocessing, as converting them to another format can be executed using a rule-based method that leverages the already existing document structure.

However, for the purpose of testing parsing methods, the regulations are converted to plain .txt formats. This is done because the raw text of the Dutch regulations contains explicit textual indicators for components, and consistent numbering. Listing 1 shows a snippet of this .txt serialization, Listing 2 shows the original .xml serialization.

Hoofdstuk 1. Algemene bepalingen
Afdeling 1.1. Algemeen
Artikel 1.1. (begripsbepalingen)
Bijlage I bevat begripsbepalingen voor de toepassing van dit besluit.
Artikel 1.1a. (grondslag)
1. Dit besluit berust op de artikelen 4.3, eerste lid, 5.1, 16.1, derde lid, en 23.1 van de wet.
2. Dit besluit berust ook op de artikelen 119 en 119a van de Woningwet.
Afdeling 1.2. Internationaalrechtelijke verplichtingen
Artikel 1.2. (wederzijdse erkenning)
Met een kwaliteitsverklaring bouw, certificaat, keuring of norm als bedoeld in dit besluit wordt gel...
Hoofdstuk 2. Algemene bepalingen voor bouwwerken
Afdeling 2.1. Algemeen
Artikel 2.1. (toepassingsbereik: activiteiten)

Listing 1 Snippet of the .txt serialization of the Dutch regulations

```
<hoofdstuk bwb-ng-variabel-deel="/Hoofdstuk1" stam-id="14709223" versie-id="25202402" id="C41798441" label-id="14661054" inwerking="2024-01-01" label="Hoofdstuk 1" bron="Stb.2018-291" effect="nieuwe-regeling" ondertekening_bron="2018-07-03" publicatie_bron="2018-08-31" publicatie_iwt="2023-04-07" status="goed">
```



```
<kop>
  <label>Hoofdstuk</label>
  <nr status="officieel">1</nr>
  <titel status="officieel">Algemene bepalingen</titel>
</kop><afdeling bwb-ng-variabel-deel="/Hoofdstuk1/Afdeling1.1" stam-
id="14709193" versie-id="27188292" id="C41788921" label-id="14661024"
inwerking="2024-01-01" label="Afdeling 1.1" bron="Stb.2020-400"
effect="structuurwijziging" ondertekening_bron="2020-09-16" publicatie_bron="2020-
10-28" publicatie_iwt="2023-04-07" status="goed">
  <kop>
    <label>Afdeling</label>
    <nr status="officieel">1.1</nr>
    <titel status="officieel">Algemeen</titel>
  </kop><artikel bwb-ng-variabel-deel="/Hoofdstuk1/Afdeling1.1/Artikel1.1" stam-
id="14709183" versie-id="27188282" id="C42580661" label-id="14661014"
inwerking="2024-01-01" label="Artikel 1.1" bron="Stb.2020-400" effect="wijziging"
ondertekening_bron="2020-09-16" publicatie_bron="2020-10-28" publicatie_iwt="2023-
04-07" status="goed">
```

Listing 2 Snippet of the .xml serialization of the Dutch regulations

4.1.1.2. Portuguese, Lithuanian, and Danish regulations

The Portuguese, Lithuanian and Danish regulations are published in PDF (and/or HTML) and have a clear visual hierarchy. For example, in the Portuguese case (Listing 3), Roman numerals are used in all-caps to indicate chapters, articles are sentence-cased with Arabic numerals and a masculine ordinal indicator, and individual clauses start with a number and an em dash. To preserve the visual information, these regulations were converted to .docx formats.

CAPÍTULO V

Evacuação de efluentes de combustão

Artigo 92.º

Condutas de evacuação de efluentes de combustão

1 – A extração dos efluentes dos aparelhos de combustão deve ser feita para o exterior do edifício por meio de condutas construídas com materiais da classe A1, que observem o disposto no artigo 31.º, e ainda que:

- a) Possuam reduzida permeabilidade;
- b) No caso de funcionarem em sobrepressão:

- i) Sendo interiores ao edifício, estejam alojadas em ducto devidamente ventilado;
- ii) Sendo exteriores ao edifício, respeitem as distâncias de segurança aos vãos abertos em fachadas e coberturas constantes dos artigos 7.º e 10.º, no que se aplicar.

Listing 3 Snippet of the Portuguese regulations

4.1.1.3. Belgian regulations

Unlike the previous cases, the Belgian regulations do not employ structured textual or visual indicators to represent the document structure. The regulations evolved mostly incrementally, and are shaped by successive layers of national norms. This caused significant variation in the visual and typographical treatment of the documents, for example, interchanging bolding, italics, and brackets. These documents are converted into Markdown (.md) using LlamaParse, leveraging manually annotated .md reference templates as multi-shot prompt examples to guide structural styling. Listing 4 shows the resulting .md serialization.

```
# **2 COMPARTIMENTERING EN EVACUATIE**
```



```
## **2.1** ***[Grootte van de compartimenten]***  
Het gebouw is verdeeld in compartimenten waarvan de oppervlakte kleiner is dan  
2500 m2, met uitzondering van de parkeergebouwen (zie 5.2).  
[...]  
## **2.2 Evacuatie van de compartimenten.**  
### **2.2.1 Aantal uitgangen.**  
Elk compartiment heeft minimaal :  
- een uitgang indien :  
  - de gebruikers zonder door het trappenhuis te moeten gaan een gevelopening  
  kunnen bereiken die toegankelijk is voor de ladders van de brandweer of indien  
  zulke opening niet bestaat moeten de gebruikers een terras, toegankelijk voor de  
  brandweer, kunnen bereiken dat ten minste 1 m2 meet, een vloer heeft met Rf 1h en  
  een leuning minimum 1 m hoog, die gedurende 1 h aan het criterium "vlamdichtheid"  
  van NBN 713-020 beantwoordt;
```

Listing 4 Snippet of the Belgian regulations in .md

4.1.1.4. Tabular and unstructured data

For documents containing unformatted tables or unstructured PDF layouts, more advance methods were applied, including the use of optical character recognition (OCR)/Tesseract. Neural network approaches (using fine-tuned BERT models) were applied to parse other unstructured PDF files.

4.1.2. Regulatory document interpretation methods

4.1.2.1. Rule-based extraction

When regulatory documents provide structured schema markups, such as in the Dutch case, deterministic rule-based algorithms can be used for parsing the regulations into an RDF format. This approach is applied to the XML representations of the Dutch building codes. A Python script is built on top of rdflib to enable this parsing. This is done by:

1. Mapping structural XML tags (such as <hoofdstuk>, <afdeling> and <artikel>) into corresponding classes,
2. Extracting metadata embedded in the XML attributes (such as labels, identifiers, version numbers, issue dates, and publication dates) and asserted as literals.
3. Structural containment is derived from the identifiers of each part of the document (e.g. "/Hoofdstuk5/Afdeling5.1" or "/Chapter5/Section5.1" indicates that Section 5.1 is contained in Chapter 5).
4. Tables are mapped cell-by-cell into graphs, using the CSVW ontology (<http://www.w3.org/ns/csvw#>) . This enables preserving cell positions, row/column relationships, and visual formatting, so that the table can be reproduced later.
5. Explicitly mentioned cross-references are automatically identified and mapped as relationships in the graph. However, implicit references ("In addition to the first paragraph...") lack the explicit information in the XML and require other methods to be extracted and parsed into the RDF graph.

4.1.2.2. Hybrid NLP-Rule-based extraction

For plain-text regulations (such as the .ttl representation of the Dutch regulations), a hybrid method is applied. The spaCy library is used to execute NLP-tasks, such as tokenization, part-of-speech tagging, dependency parsing, and token-based matching (using spaCy's attribute_ruler). This is used to extract rule-based token patterns to identify structural components from the text, as well as cross-references. GLiNER (Generalist and Lightweight Model for Named Entity Recognition) is then used to provide neural Named Entity Recognition capabilities. Both operate in the same spaCy pipeline.

The attribute_ruler is used to define certain patterns that denote hierarchical elements, such as by using letters, roman numbers, linestarts, and so forth. After tagging these hierarchical elements in the document using the various patterns, a rule-based hierarchical level is assigned, resulting in a tree-like hierarchical distribution of



those components. The assumption is that this hierarchy is fixed and known for a specific document (e.g., all paragraphs are always in a lower hierarchical level as the chapters).

GLiNER is then used to label parts of the document with custom labels, such as ‘standard’ for ‘NEN-EN 13501-1’. This is an out of the box approach and does not require training. Arguably, LLMs could do a similar or even better job at the time of writing, but the conceptual architecture stays similar. Then, rule-based components were added to GLiNER to represent patterns for cross-references. These include coded references (“4.12a”), structural references (“Chapter 5”), ordinals (“first, second”), relative references (“this” or “current”) and conjunctions combining those elements (“and”, “or”, “up to and including”).

4.1.2.3. Multi-stage LLM prompting

Due to inconsistencies in the Belgian regulations regarding style, numbering, and formatting, identifying structural components through fixed rules was considered insufficiently robust, while developing a dedicated hybrid model would require substantial training effort. Instead, an out-of-the-box LLM, Anthropic’s Claude 3.5 Sonnet, was used for the final interpretation stage.

After conversion to Markdown, the regulations were segmented through regular-expression pattern matching of hierarchical numerical headings (e.g., “# 1”, “## 1.2”, “# 1.2.3”). The resulting sections were represented as hierarchical Section instances, with parent–child relationships derived from their numbering.

Each section was then processed using a multi-shot prompt containing the FireBIM ontology as a controlled vocabulary, transformation rules for generating instance data, and example mappings from legal text to TTL. The LLM was instructed to identify atomic legal requirements, represent them using ontology concepts, avoid duplication across hierarchical levels, and enrich relevant literals with ontology-based HTML hyperlinks for thematic tagging.

The generated TTL was cleaned of formatting artefacts and subjected to syntax and consistency checks, including verification against the previously extracted section numbering. Valid outputs were incorporated into the knowledge graph. Failed validations triggered an automated retry in which the previous error trace was provided as corrective feedback, allowing the LLM to iteratively correct its output.

4.1.3. Semantic structuring of the documents using the FireBIM Regulation Ontology

Regardless of whether rule-based, NLP, LLM, or hybrid interpretation is used, all extracted regulatory data is standardized into an RDF graph representation using the FireBIM Regulation Ontology. The conversion procedure from regulations in natural language to the National Regulation Graph (in RDF) makes use of the FireBIM Regulation Ontology. This ontology (see Figure 6) describes semantic concepts to structure the regulations from various countries in a machine-understandable structure.

FRO provides a country-independent vocabulary to represent the structure of legal documents, without enforcing a specific hierarchy. Some foundational concepts include:

1. fro:Authority: The legal body issuing regulations within a given jurisdiction,
2. fro:Jurisdiction: The political or geographic domain where the document is legally binding,
3. fro:Document: The (regulatory) document, such as national building codes,
4. fro:DocumentSubdivision: an overarching class for structural components, with subclasses including fro:Chapter, fro:Section, fro:Paragraph, fro:Article, fro:Statement, fro:Table, fro:Figure, and others,
5. fro:Reference and fro:Topic: classes capturing internal/external document or thematic references.

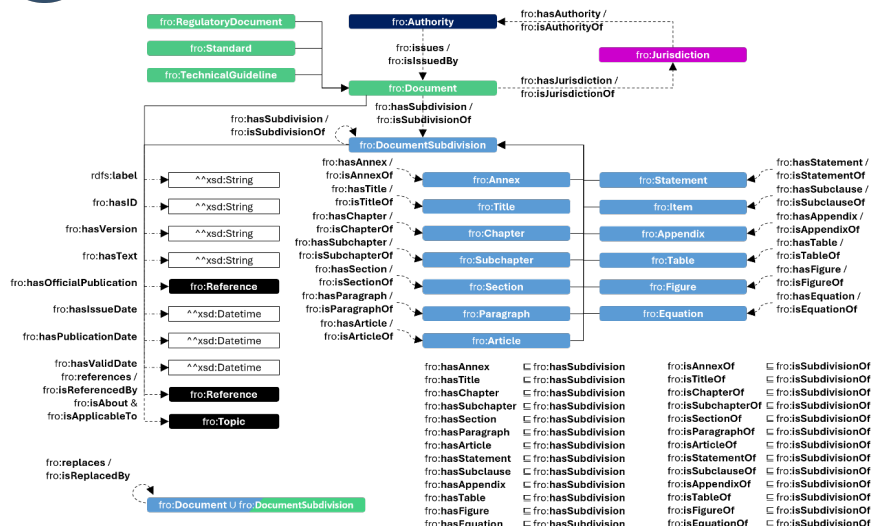
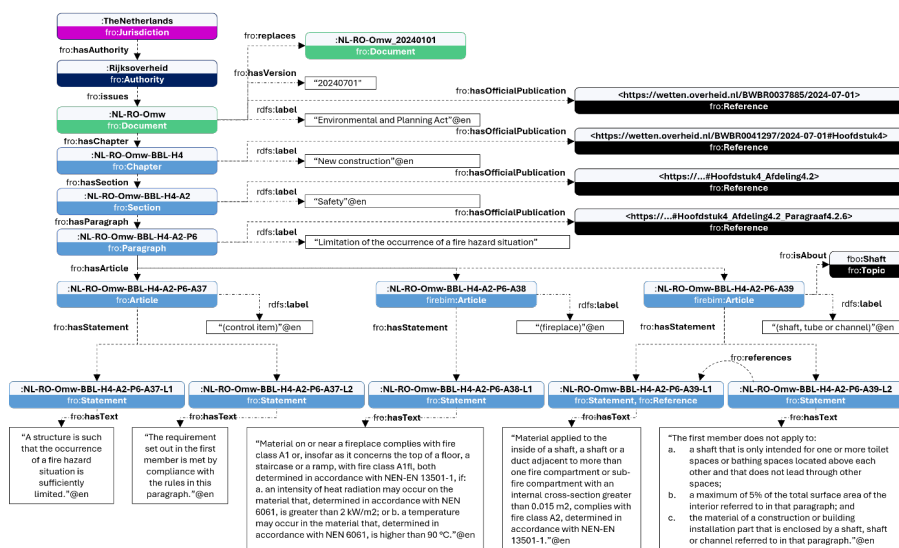


Figure 7 and Listing 5 show how a chapter of the Dutch building code could be serialized as RDF. The ontology enables including identifiers, labels in multiple languages, and a reference to an external source, such as the original web location of the regulation. The regulatory articles themselves in this stage still appear as literals in the graph.



inst:NLR00mWBBLH4



```
a      fro:Chapter ;
fro:hasID      "NLROOMWBLH4" ;
rdfs:label    "Hoofdstuk 4: Nieuwbouw"@nl ;
rdfs:label    "Chapter 4: New construction"@en ;
fro:issued    "2024-01-01" ;
rfo:hasOfficialPublication <https://wetten.overheid.nl/
                        BWBR0041297/2024-01-01#Hoofdstuk4> .
```

Listing 5 An RDF description of a chapter of the Dutch building code

Multiple software implementations have been built to achieve the generation of these RDF-based regulation graphs. While these implementations all need a slightly different interface with the preprocessing implementations (that are different for each country, or even document), the subsequent steps follow the following sequence:

1. Input acquisition: Ingesting semi-structured data outputs from the preprocessing and interpretation phase,
2. Semantic parsing: Identifying entities and assigning their respective class from the FRO ontology,
3. URI construction: Mining HTTP URIs for every entity,
4. Triple creation: Creating subject-predicate-object triples following the FRO ontology using rdflib,
5. Hierarchy creation: Creating triples with parent-child relationships to establish the document's hierarchical structure,
6. Metadata enrichment: Attaching literals with metadata, such as the human-readable text of an entity, a name, an ID, and version info,
7. Quality validation: Checking the resulting RDF syntax,
8. Serialization: Exporting the knowledge graph into the Turtle (.ttl) format.

4.1.4. Human-readable formats

Since RDF graphs could be difficult to evaluate, a deterministic graph-to-HTML script is created. The script enables visualizing the RDF graph-based regulations in an HTML format, such that an expert can visually spot the differences between the source document and the HTML. Since the structuring of the RDF graph is done using a country-independent ontology, the HTML generation method is country-independent. An example of the source regulations in PDF and the resulting HTML representation in Belgium can be seen in Figure 8. Figure 9 shows how an HTML representation was made for the Portuguese regulations. This HTML page is interactive, meaning that one could interact with the terms within the regulations to find their definitions (using the content of the ontologies).

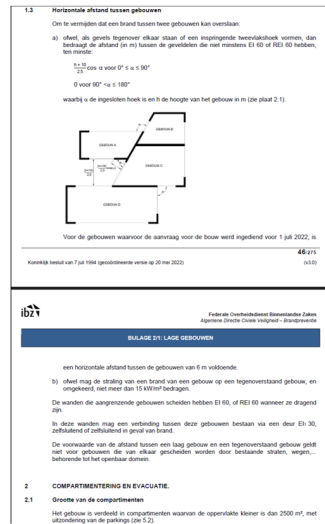
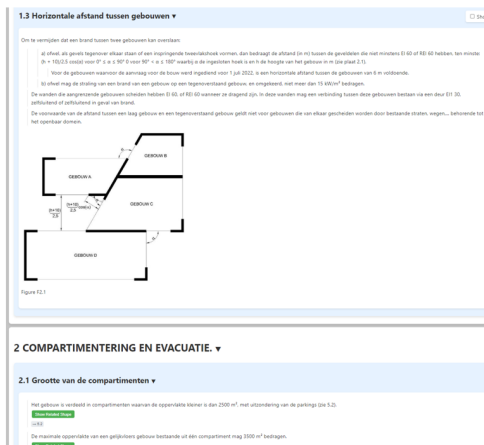


Figure 8 Visual comparison of generated HTML (left) and original PDF (right)

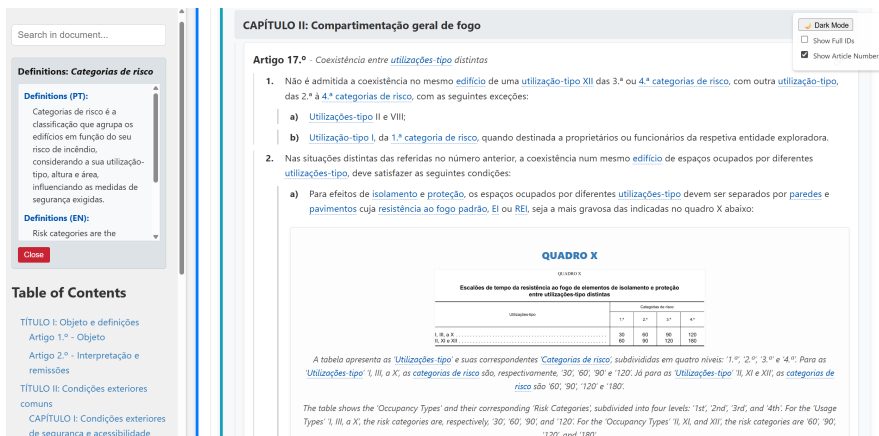


Figure 9 Interactive HTML representation of the Regulation graph in Portugal

4.2. Rule interpretation and digital rule generation

4.2.1. RASE

The first approach for rule interpretation is a manual-tagging-based approach using the RASE tagging method. RASE stands for Requirement, Applicability, Selection, and Exception and is a HTML-based tagging method, where (fire safety) experts color-code sentences based on these four tags.

The RASE tagging activities started with an internal training workshop. The methodology and the four colors were explained:

ITEA4 22003 **FireBIM**
D2.3

[CONFIDENTIAL]

30 / 55
[DRAFT]



- Applicability (green) defines the scope or context where the rule applies (e.g. a building type),
- Requirement (blue) defines the normative condition or metric that must be met.
- Selection (red) defines inclusive options for the applicability.
- Exception (yellow) defines exclusionary criteria or alternative compliance paths.

Fire safety experts from the multiple participating countries then applied this method on their national regulations. A practical example on the Danish BR18 fire safety regulations is shown below. Figure 10 shows a specific Danish fire regulation with the applied color coding.

In buildings where the floor on the top floor is no more than 12.0 m above ground level, building parts that delimit a fire section can either be constructed with a fire resistance of at least:

1. Building part class EI 60 / A2-s1,d0 [BS building part 60] or
2. Building part class EI 60 / D-s2,d2 [BD building part 60] [class B material] constructed with at least cladding class K2 60 / A2-s1,d0 [60 minute fire protection system], or
3. Building part class EI 60 / D-s2,d2 [BD building part 60] [class B material] constructed with at least fire-protective cladding made of material class A2-s1,d0, and this is documented, cf. DS/EN 1995-1-2, that the wood does not burn through in the first 60 minutes of a standard fire exposure

In the case of floor separations, the protection may be limited to a fire below the floor separation.

Requirement
Applicability
Selection
Exception

Figure 10 A RASE-tagged Danish fire regulation

Such representation can then be mapped to an OPCTU (Object, Property, Comparison, Target, Unit) table. Such a table can be used to express how the normative regulation maps to specific objects, properties, and target(s) (values). Table 2 shows an example of such a table, for the regulation in Figure 10. The RASE tags can also be converted to procedural code, a tree, and a graph representation.

Table 2 OPCTU table of the Danish fire regulation

	Metric phrase	Type	Object	Property	Comparison	Target	Unit
1	buildings	applicability	fbo:Floor	owl:inverseOf fbo:hasElement	Equals	Fbo:Building	
2	floor	applicability	:Thing	Rdf:type	Equals	Fbo:Floor	
3	floor on the top floor	applicability	fbo:Floor	owl:inverseOf fbo:hasElement	Equals	fbo:TopStorey	
4	floor [...] is no more than 12.0 m above ground level	applicability	fbo:Floor	fbo:elevation	Less or equal then	12.0	Meters
5	building parts	applicability	:Thing	rdf:type	equals	fbo:Element	



6	delimit a fire section	applicability	:Thing	fbo:delimitsFireSection	equals	True	
7	Building part class EI 60 / A2-s1,d0 [BS building part 60]	requirement	fbo:Element	fbo:fireRating	more or equal then	EI 60 / D-s2,d0	Minutes
8	Building part class EI 60 / D-s2,d2 [BD building part 60] [class B material]	requirement	fbo:Element	fbo:fireRating	More or equal then	EI 60 / D-s2,d2	Minutes
9	at least cladding class K2 60 / A2-s1,d0 [60 minute fire protection system]	requirement	fbo:Cladding	fbo:fireRating	More or equal then	A2-s1,d0	Minutes
10	Building part class EI 60 / D-s2,d2 [BD building part 60] [class B material]	requirement	fbo:Element	fbo:fireRating	More or equal then	EI 60 / D-s2,d2	Minutes
...							

The OPCTU table would also enable to draft a SHACL shape representing that regulation. Listing 6 shows a conversion of the OPCTU to SHACL. The bold numbers in the SHACL shape indicate the lines of the OPCTU table.

```
@prefix fbo: <http://example.org/ontology#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
fbo:DanishRegulationShape a sh:NodeShape ; 16
    sh:target [
        a sh:SPARQLTarget ;
        sh:prefixes fbo: ;
        sh:select """
            SELECT ?this
            WHERE {
                ?this rdf:type fbo:Element ; 5
                ?building rdf:type fbo:Building ; 1
                ?floor rdf:type fbo:Floor ; 2
                ?topFloor rdf:type fbo:TopStorey ;
```



```

    ?topFloor ^fbo:hasElement ?floor ; 3
    ?floor fbo:floorElevation ?elevation
    FILTER (?elevation <= 12.0) ; 4
    ?this fbo:delimitsFireSection True . 6
  }
  "" ;
sh:or (
  [ sh:property [
    sh:path fbo:fireRating ;
    sh:minInclusive "EI 60 / D-s2,d0" ;
  ] 7
  [ sh:and (
    [ sh:property [
      sh:path fbo:fireRating ;
      sh:minInclusive "EI 60 / D-s2,d2"
    ] 8
    [ sh:property [
      sh:path fbo:hasCladding |
      sh:minInclusive "A2-s1,d0" ;
    ] 9
    ] )
  ] 17
  [ sh:and (
    [ sh:property [
      sh:path fbo:fireRating ;
      sh:minInclusive "EI 60 / D-s2,d2"
    ] 10
    [ sh:property [
      sh:path fbo:hasCladding |
      sh:minInclusive "A2-s1,d2" ;
    ] 11
    [ sh:and ( [ sh:property [
      sh:path
      sh:class fbo:Wood ;
    ] 12
    [ sh:property [
      sh:path fbo:fireRating
      sh:minInclusive "60" ;
    ] 13
    ] )
    ] )
  ] 18
  [ sh:and (
    [ sh:property [
      sh:path rdf:type ;
      sh:class fbo:FloorSeparation ;
    ] 14

```



```
    ]  
    [ sh:property [  
      sh:path fbo:fireProtection ;  
      ?????????????????? ;  
    ] 15  
  ] )  
)  
;
```

Listing 6 SHACL shape representing the RASE-tagged Danish regulation

4.2.2. NLP-based interpretation pipeline

This project also developed an NLP-based regulation converter, that takes natural language regulations as input and outputs SHACL shapes. The converter uses a rule-based procedure (see Figure 11) based on POS tagging, dependency parsing, and noun chunk generation in spaCy. This rule-based procedure is, other than the LLM-based procedure in the previous paragraph, not such a black-box, enabling evaluation of why a certain shape was created and how it was populated. The full procedure of how this converter converts natural language to SHACL shapes is presented in Donkers & Petrova (2024).

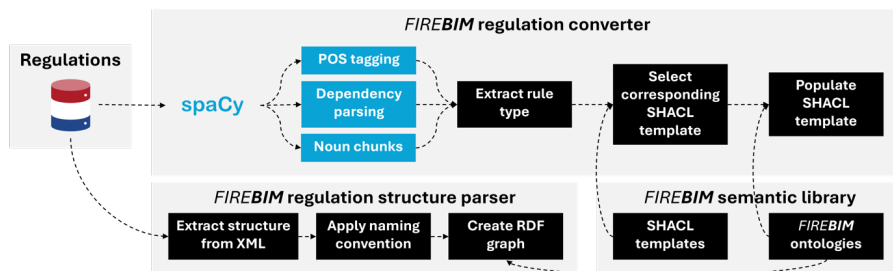


Figure 11 NLP-based pipeline to generate SHACL shapes from (Dutch fire safety) regulations

Figure 12 shows the result of the spaCy procedure for a Dutch fire safety regulation. Based on these results, a stepwise procedure is followed to convert the sentence to a SHACL shape. The steps in this procedure are as follows:

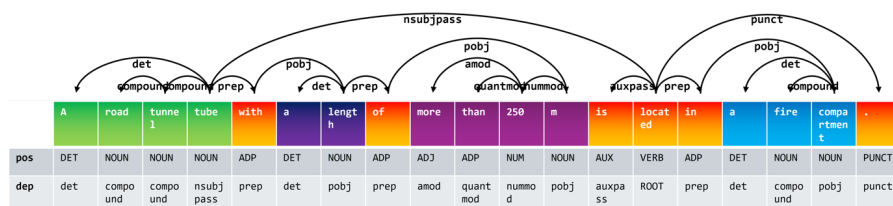


Figure 12 Dependency tree and noun chunks of a Dutch fire safety regulation

1. Based on the dependency parsing, the root word of the sentence is extracted ('located').
2. The group of words that the root is directly related to, such as auxiliary clauses ('AUX') or adpositions ('ADP') are extracted and grouped with the root ('is located in'). This group of words is used to determine the rule type based on a mapping table that was created for this project.



3. The nominal subject (or equivalent, such as the passive nominal subject) that is the child of the root is extracted (*'area'*).
4. The noun chunk of the nominal subject is created (*'An occupied area'*), and in this noun chunk, it is checked whether there are any special determiners (*'An'*), such as quantifiers.
5. All child relations of the nominal subject that modify the nominal subject, such as verbs, adjectival modifiers, or noun compounds are extracted (*'occupied'*).
6. These child relationships are combined with the nominal subject (*'occupied area'*).
7. Based on the results of step 2, step 7 and onwards changes. For this regulation, the other nouns that are children of the root are extracted (*'compartment'*).
8. Steps 4 and 5 are repeated, but now for this new noun (*'protected, sub, -, fire'*).
9. The child relationships are combined with the noun (*'protected sub-fire compartment'*).

Listing 7 shows the resulting SHACL shape after following this stepwise procedure.

```
:NLROBBLH4A2P8A50L3
  a
    rdfs:comment      sh:NodeShape , fro:Statement ;
                      "A road tunnel tube with a length of more than 250 m is
                      located in a fire compartment."@en ;
    sh:target         [
                        a          sh:SPARQLTarget;
                        sh:prefixes fbo: , bop: , quantitykind: ,
bot: ;
                        sh:select  """ SELECT ?target
                                WHERE {
                                  ?target a fbo:RoadTunnelTube ;
                                  fbo:length ?value ;
                                  FILTER (?value > 250) . """ ;]
    sh:property [
      sh:path      fbo:isLocatedIn ;
      sh:class     fbo:FireCompartment ; ]
```

Listing 7 Resulting SHACL shape of the Dutch fire safety regulation

4.2.3. Expert-driven diagram-to-SHACL workflow

Kommenterede [AD3]: @Tomas

4.2.4. Semi-automatic diagram-to-SHACL workflow

Next to manually creating the intermediate diagrams, FireBIM also developed a method to semi-automatically generate them (Figure 13). Sections of the regulatory document, such as articles or individual statements, are fed to an LLM to create flowcharts representing the rule in a pseudo-code format. A text-based diagramming format (Mermaid) is used to generate both human- and machine-interpretable diagrams.

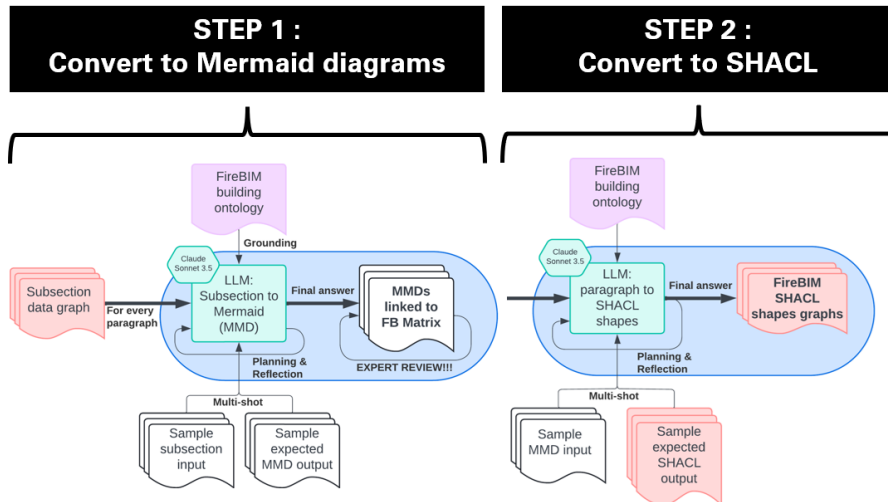


Figure 13 The workflow discussed in section 5.2

These diagrams are easily checked and compared to the source legislation by fire experts, giving us an extra human-in-the-loop validation possibility to break up the large jump from the non-interpreted text in the paragraph graphs to fully machine-readable SHACL shapes. Important here is that we force the LLM to maintain connections to the ontology, having it include the URIs for concepts where possible, to avoid confusion on any definitions. An example of a generated flowchart is shown in Figure 14.

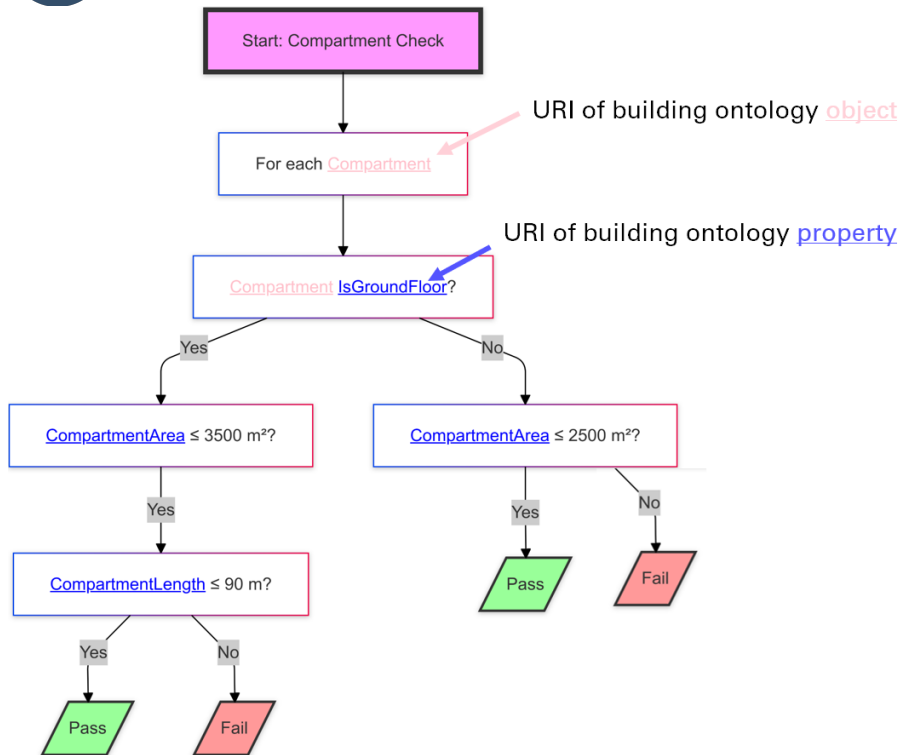


Figure 14 An example of a generated flowchart

4.2.5. Neuro-symbolic AI method

Finally, a neuro-symbolic AI method is developed to interpret regulatory statements and generate machine-interpretable rules in the form of SHACL shapes. The method combines neural language processing with symbolic knowledge representations to ground the interpretation of regulatory text in domain ontologies and formal rule structures. As shown in Figure 15, the pipeline transforms a regulatory statement through a sequence of linguistic preprocessing, ontology-based semantic grounding, formalisation in First-Order Logic (FOL), and SHACL generation. The intermediate representations are retained to support human validation of the generated rules.

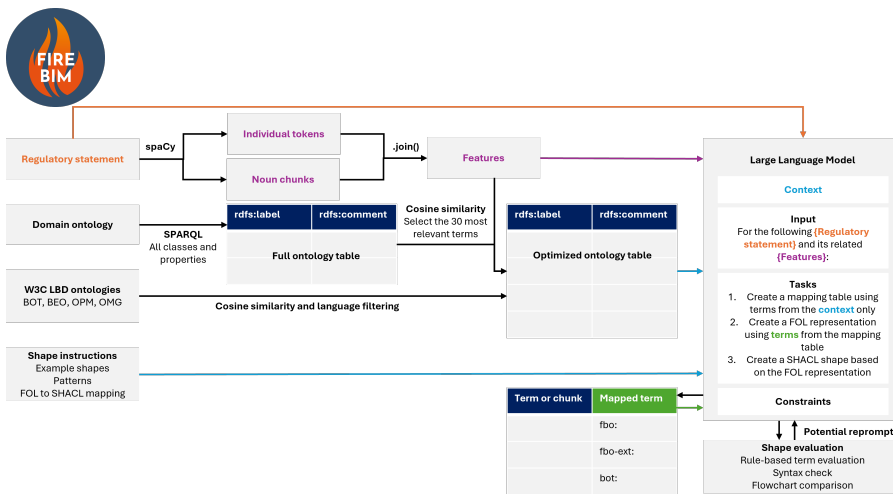


Figure 15 A neuro-symbolic pipeline for generating SHACL shapes

4.2.5.1. Linguistic preprocessing

The pipeline starts from an individual regulatory statement or a selected set of statements. The text is analysed to extract linguistic features, specifically the individual tokens and noun chunks, using spaCy. Noun chunks are important for domain-specific expressions in which the meaning of a noun depends on its modifiers. For example, a “portable fire extinguisher” should be mapped as a concept rather than split into separate terms. It should also be known that “portable” is a modifier of “extinguisher”, and not of “fire” (because it’s not an extinguisher for portable fires). The resulting feature set forms the input for the interpretation stage.

4.2.5.2. Inserting symbolic knowledge in the context

Then, ontologies are loaded. These include a domain ontology, in this case the FireBIM Building Ontology (FBO), and potentially other ontologies (such as the W3C LBD ontologies). The FBO ontology provides fire-safety-related terminology, whereas the LBD ontologies provide more general concepts for building topology, elements, properties, and geometry.

Directly loading these ontologies in the context of the LLM is inefficient because of their (token) size. The implementation therefore reduces the size of the ontology before prompting. Multilingual descriptions are first filtered out, after which a cosine similarity algorithm is applied against the embedded ontology and the embedded features. The sentence-transformers/all-mpnet-base-v2 model with Mean Pooling was used for the embeddings, and sklearn.metrics.pairwise.cosine_similarity is used to compute the cosine similarity scores. The most relevant terms are retained and loaded in the context of the LLM, leading to significant token reductions.

Next to ontologies, the method also loads a set of shape instructions in the context of the LLM. These include example shapes (in a textual rule-SHACL shape example sets), desired patterns (in pattern-SHACL shape sets), and FOL to SHACL mapping (explaining how certain FOL logic should be represented in SHACL shapes).

4.2.5.3. Formalisation of explicit regulatory logic

Regulatory statements are interpreted as logical rules first (in First-Order Logic), rather than directly converting them to SHACL. This intermediate representation preserves the logical structure of the regulation while remaining independent of the eventual rule representation. Laws are naturally structured as logical implications, where:

$$\forall x(\text{Antecedent}(x) \rightarrow \text{Consequent}(x))$$

First-Order Logic (FOL) follows that mapping and also splits the antecedent and the consequent. The FOL logic can be mapped to RASE tags (where the antecedent would map to Applicability/Selection and the consequent to Requirement/Exception). Discussion on whether Exceptions should be modelled in the right-hand side of the equation exist, and one could either model them as negation ($A \wedge \neg E \rightarrow R$ (making it part of the antecedent) or



as a disjunction to the requirement $A \rightarrow (R \vee E)$ (making it part of the consequent). The execution of that logic should result in the same outcome, however, when modeling this logic to SHACL shapes later, placing exceptions in the `sh:PropertyShapes` rather than in the `sh:Target` does influence the practical execution of these shapes (and their performance). The pipeline currently assumes $A \rightarrow (R \vee E)$.

4.2.5.4. Transformation into SHACL shapes

The formalised rule is then transformed into a SHACL shape, which provides the executable representation of the regulation. This SHACL shape follows a predefined modelling pattern in which the logical components of the FOL are mapped to corresponding SHACL constructs. The antecedent of the rule determines the target of the SHACL shape (using `sh:SPARQLTarget`), and the consequent is represented by one or more property shapes that specify the requirements that the target must satisfy. Depending on the type of regulation, the property shapes can express required properties, cardinalities, relationships between elements, value restrictions, mathematical constraints, and logical combinations of these requirements. More complex requirements can be represented by nesting node shapes and logical operators. However, the general aim is to create relatively atomic SHACL shapes that can later be combined to represent more complex rules.

A set of predefined patterns and modeling idioms for certain recurring constructs, such as property constraints, inverse relationships, cardinalities, topological relationships, and logical operators, constrains the SHACL generation to reduce inconsistencies between shapes. This ensures that SHACL shapes can be evaluated and executed using similar methods.

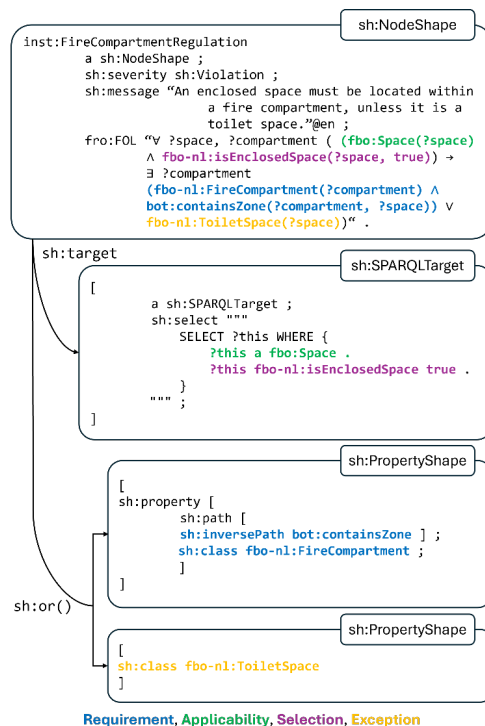


Figure 16 A resulting SHACL shape



Figure 16 shows a resulting SHACL shape. The `sh:target` identifies the entities for which the regulation is applicable (“enclosed space”) and the property shape expresses the requirement that must be satisfied, as well as an exception. The metadata of the SHACL shape contains both the original statement in natural language, and the intermediate FOL representation. This way, other non-SHACL methods could build upon this work if needed.

4.2.5.5. Human-readable representation and validation

To support human (fire expert) validation of the generated SHACL shape, the shape is transformed into a human-readable flowchart (Figure 17). The flowchart presents the logical structure of the rule as a sequence of conditions and requirements, allowing domain experts to ‘read’ the logic that the SHACL shape follows.

The flowchart is generated directly from the SHACL shape using a rule-based transformation. This ensures that the human-readable representation reflects the same machine-interpretable rule that will be used for compliance checking. References to the ontology are included in the diagram, allowing fire safety experts to not only follow the sequence but to also check whether the ontology terms are correctly mapped.

The flowchart therefore provides an additional validation layer between the original regulatory text and the SHACL shape. This supports a human-in-the-loop approach in which the LLM performs the initial formalisation, while domain experts remain responsible for validating the digital rule.

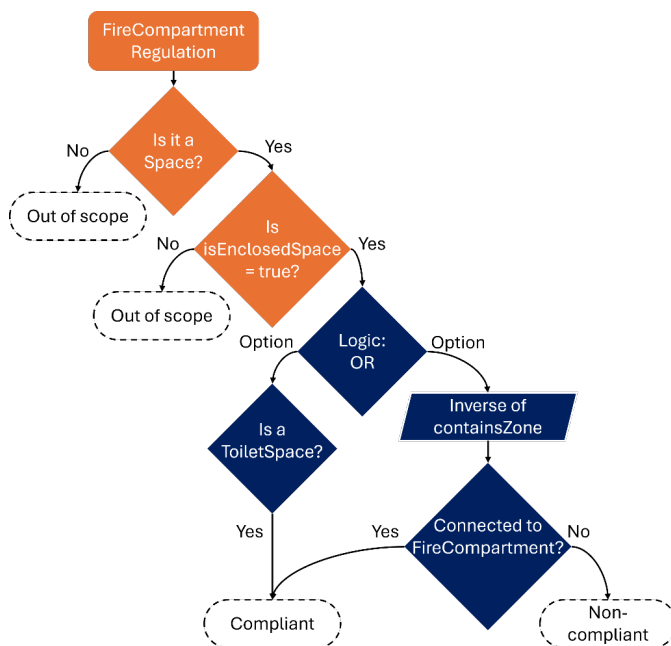


Figure 17 Flowchart representation of a fire safety regulation

Next to the SHACL-to-diagram conversion, another script enables converting the SHACL shape to a text-based description of the SHACL logic (Figure 18). Similar to the diagram, this result can be used by fire safety experts to evaluate the internal logic of the SHACL shape.



```
fbbl:GroundFloorCompartmentShape
a sh:NodeShape ;
sh:targetClass fbbl:Compartment ;
fbbl:flowchartNodeID "E", "G" ;
firebim:rulesource firebim:Section_2_1 ;
sh:or(
  [
    sh:and (
      [
        sh:property [
          sh:path fbbl:IsGroundFloor ;
          sh:hasValue true ;
        ]
      ]
    )
  ]
  [
    sh:property [
      sh:path fbbl:CompartmentArea ;
      sh:maxInclusive 3500 ;
      sh:datatype xsd:decimal ;
    ] ;
    sh:property [
      sh:path fbbl:CompartmentLength ;
      sh:maxInclusive 90 ;
      sh:datatype xsd:decimal ;
    ]
  ]
)
]
]
) .
```

SHACL



For every Compartment:
AT LEAST ONE of the following must be true:
- ALL of these must be true:
• The IsGroundFloor must be exactly true.
• The CompartmentArea must be at most 3500.
• The CompartmentLength must be at most 90.
- The CompartmentArea must be at most 2500.

Text

Figure 18 An example of automatic translation of SHACL (left) to text (right)

4.2.5.6. Large Language Model prompt strategy

A Large Language Model (Gemini, via Google's genai Python library) is used to execute a multi-stage prompting strategy. First, the context (including the optimized ontologies and the shape instructions), constraints, and the input data (the regulatory text and the extracted features) are loaded. Then, a set of tasks is given. The LLM subsequently:

1. Creates a mapping table between the extracted features and terms from the optimized ontology only,
2. Creates a FOL representation using terms from the mapping table only,
3. Creates a SHACL shape based on the FOL representation.

This decomposition separates the (semantic) interpretation, logical formalisation, and transformation to the SHACL shape. The intermediate mapping table and FOL are retained alongside the generated SHACL shape, allowing the interpretation to be validated.

4.3. Building graph generation and geometric computations

Translation of native building models into graph-based representations is a critical step in enabling SHACL-based automated compliance checking. This step involves converting standard exchange formats, for example, IFC, into RDF graphs. Specific for fire safety rule checking is the conversion of complex geometric constructs.

4.3.1. IFC to LBD conversion

A primary barrier to using RDF-based building information models is the efficient transformation from IFC files into RDF graphs. To address this issue, IFC2LBD-Neo was recently developed, which is a Rust-based converter designed for efficient Linked Building Data (LBD) generation (Figure 19). The system employs a staged pipeline architecture that supports parallel processing, allowing for highly scalable IFC to RDF conversions. The conversion software supports IFC2x3 up to IFC4x3. To accommodate different pipelines, the serializers can generate either merged Turtle graphs, separate Turtle graphs per parallel pipeline, or chunked N-Quads. These chunked N-Quads split generated graphs by line count or byte thresholds, facilitating the faster incremental loading of large RDF graphs into triple stores.

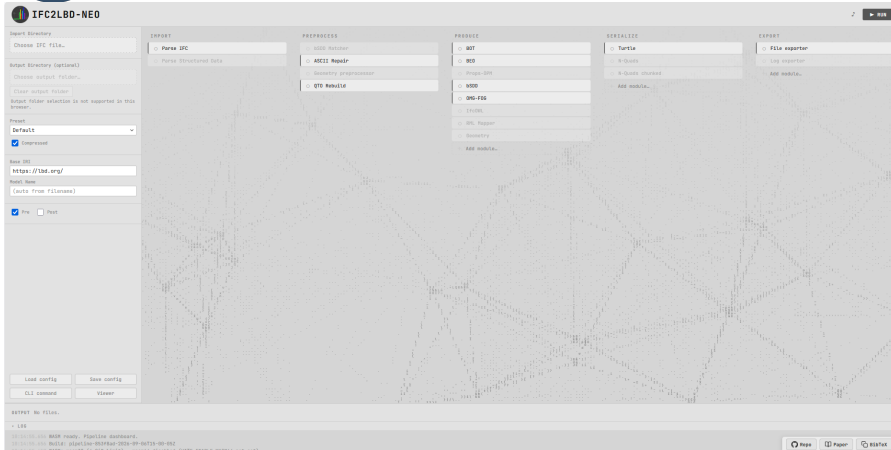


Figure 19 An overview of the IFC2LBD-Neo converter

4.3.2. Procedures for integrating geometric functionalities

Because the converted RDF graphs typically lack topological and geometric details, FireBIM developed methods to enhance these graphs with this information. A Python-based framework is developed that integrates geometric processing capabilities of IfcOpenShell directly with the LBD graphs.

The method follows a four-step conceptual architecture:

1. Configuration setup: a configuration file defines a SPARQL query and points to the target LBD graph, which must contain references to the source IFC file and the global unique identifiers (GUIDs) for the target nodes, such as elements or conceptual zones like fire compartments
2. Querying: An RDFlib module queries the specified LBD graph to extract a list of element or zone URIs alongside their corresponding GUIDs
3. Geometric processing: the IfcOpenShell library takes the extracted GUIDs to locate the exact IfcElements and IfcZones within the source IFC model and executes the required geometric calculations. These calculations are predefined and available in a library.
4. Graph enrichment: The calculated geometric results are converted into new RDF triples using GeoSPARQL 1.1, and injected back into the original LBD graph.

A list of functions is developed, including simple geometric calculations (including length, width, height, area, volume, bounding box centroids, and top or bottom elevations), complex spatial calculations (including the Euclidean distance between bounding box centroids of specific elements, and clashing (using Bounding Volume Hierarchy trees, the system clashes between element-element, element-zone, or zone-zone).

4.4. Demonstrators

This section outlines four demonstrator applications that apply the digital regulations generated using this framework. Each subsection highlights a specific use case where building information is validated against different regulations using SHACL.

4.4.1. Geometric compliance checking

The first demonstrator evaluates the integration of a geometric processing library within a fire safety compliance workflow, utilizing the OpenSmartHome building as a case study. The model's topological data was extracted into a Turtle (TTL) file and semantically enriched with fire safety classifications from the FireBIM Building Ontology.



The validation focused on Article 4.50 of the Dutch building regulations, which mandates that enclosed spaces must reside within a fire compartment, with specific exceptions for technical rooms and sanitary spaces. This rule was formalized into a SHACL shape (Listing 8).

```
firebim:NLR0BBLH4A2P6A50 a sh:NodeShape ;
  sh:targetClass bot:Space ;
  sh:message "An enclosed space must be located in a fire compartment." ;
  sh:or (
    [ sh:property [
      sh:path geo:sfIntersects ;
      sh:class firebim:FireCompartment ;
      sh:minCount 1 ;
      sh:severity sh:Violation ;
    ] ; ]
    [ sh:path rdf:type ; sh:hasValue firebim:Toilet ]
    [ sh:path rdf:type ; sh:hasValue firebim:Bathroom ]
    [ sh:and (
      [ sh:path rdf:type ; sh:hasValue firebim:ElevatorShaft ]
      [ sh:path firebim:hasFireClass ; sh:hasValue "B" ]
      [ sh:path firebim:hasSmokeClass ; sh:hasValue "s2" ]
    ) ]
    [ sh:and (
      [ sh:path rdf:type ; sh:hasValue firebim:TechnicalRoom ]
      [ sh:path firebim:hasUsageArea ; sh:maxExclusive 50 ]
      sh:SPARQL [
        a sh:SPARQLConstraint ;
        sh:select """
        SELECT (SUM(?nominalLoad) AS ?totalNominalLoad)
        WHERE {
          ?room rdf:type firebim:TechnicalRoom .
          ?room bot:hasElement ?combustionEngine .
          ?combustionEngine rdf:type firebim:CombustionEngine .
          ?combustionEngine firebim:hasNominalLoad ?nominalLoad .
        }
        HAVING (SUM(?nominalLoad) <= 130)
        """ ;
      ]
    ) ]
  ) ] ;
.
```

Listing 8 SHACL shape of a Dutch fire regulation

A SPARQL query retrieved the relevant spaces and fire compartments, enabling a Python-based geometric engine to calculate spatial intersections and zone areas. These computed geometric relationships were subsequently appended to the RDF graph. The SHACL shape was then used to check the building's RDF graph for compliance with the regulation. To verify the system's robustness, the fire compartment's geometry was intentionally altered to induce a non-compliance scenario, which the SHACL validation successfully detected and flagged in the resulting report (Listing 9).

```
[
  a sh:ValidationResult ;
  sh:resultSeverity sh:Violation ;
  sh:sourceConstraintComponent sh:OrConstraintComponent ;
  sh:sourceShape firebim:NLR0BBLH4A2P6A50 ;
  sh:focusNode OF:LivingRoom ;
  sh:value OF:LivingRoom ;
  sh:resultMessage "An enclosed space must be located in a fire
compartment." ;
] .
```



Listing 9 Validation report for a space that is deliberately not located in a compartment

4.4.2. Geometric compliance checking of heterogeneous building information

The second demonstrator applies the compliance checking methodology to the Tribune demonstrator project, focusing on the intersection of MEP (Mechanical, Electrical, and Plumbing) elements and fire compartment boundaries. Structural, fire barrier, and HVAC models in IFC format were converted into RDF graphs using the IFCtoLBD converter. The system evaluated a regulatory requirement stipulating that pipe segments penetrating a compartment boundary wall must possess adequate fire resistance or feature additional sealants. A SPARQL query identified the relevant pipe segments and boundary walls (Listing 10) and the geometric engine computed the precise physical intersections between them, recording the results in the RDF graph (Listing 11).

```
1 SELECT ?PipeSegment ?guidCJB ?CompartmentBoundary ?guidS
2 WHERE {
3     ?PipeSegment rdf:type firebim:PipeSegment .
4     ?PipeSegment bpt:hasGlobalIdIfcRoot ?guidP .
5     ?CompartmentBoundary rdf:type bot:Element, beo:Wall .
6     ?CompartmentBoundary firbim:hasFireResistance "60" .
7     ?CompartmentBoundary bpt:hasGlobalIdIfcRoot ?guidC . }
```

Listing 10 SPARQL query returning all instances of pipe segments that go through compartment boundary walls

```
1 SDK:Wall_001      a          bot:Element ,
2      beo:Wall ;
3      firebim:hasFireResistance      "60" ;
4      geo:hasIntersection
5      _:Ncf7f520b7ba0406e96f38c20f12c5b52 .
6 SDK:PipeSegment_001 a          firebim:PipeSegment ,
7      geo:hasIntersection
8      _:Ncf7f520b7ba0406e96f38c20f12c5b52 .
9 _:Ncf7f520b7ba0406e96f38c20f12c5b52 geo:intersectionType "pierce" ;
10      geo:startCoordinate      "POINT (0.124 2.533 16.234)" ;
11      geo:endCoordinate      "POINT (0.124 2.833 16.234)" ;
```

Listing 11 RDF snippet of the intersection between a compartment boundary wall and a pipe segment

During the SHACL validation process, a specific pipe segment failed compliance because it lacked the required fire class data and additional sealants. Following the identification of this violation, a Large Language Model (GPT-4.0), augmented with a product catalog of fire-resistant pipe segments, successfully recommended an appropriate product solution to resolve the design error. The element that failed the compliance check could also be highlighted in a BIM viewer (Figure 20).

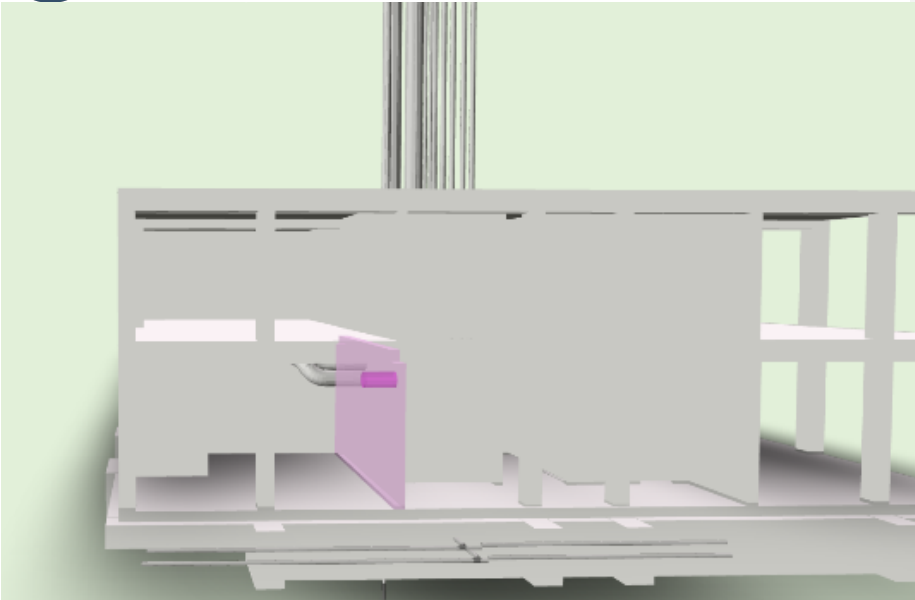


Figure 20 Visualization of a pipe segment that goes through a compartment boundary wall

4.4.3. Spatial containment and quantity checks

The third demonstrator focuses on the validation of SHACL shapes using the pySHACL engine. The evaluation targeted Article 4.51(2) of the Dutch building code, a mathematical constraint dictating that a fire compartment may contain a maximum of four caravans or ancillary functions, and their combined usable area must not exceed 1,000 square meters. To test the efficacy of the digital rule, a synthetic data graph was constructed to deliberately violate both the cardinality and area constraints (Listing 12).

```
inst:InvalidCompartment a fbo-nl:FireCompartment ;
  bot:containsZone inst:C1, inst:C2, inst:C3, inst:C4, inst:A1 .
inst:C1 a fbo-nl:ResidentialFunctionCaravan ;
  fbo:hasUsableArea 250 .
inst:C2 a fbo-nl:ResidentialFunctionCaravan ;
  fbo:hasUsableArea 250 .
inst:C3 a fbo-nl:ResidentialFunctionCaravan ;
  fbo:hasUsableArea 250 .
inst:C4 a fbo-nl:ResidentialFunctionCaravan ;
  fbo:hasUsableArea 250 .
inst:A1 a fbo-ext:AncillaryFunction ;
  fbo:hasUsableArea 200 .
```

Listing 12 Data graph that violates article 4.51(2)

The mocked fire compartment contained five relevant functional units (four caravans and one ancillary function) with a cumulative area of 1,200 square meters. When processed by the validation engine, the system accurately identified both distinct breaches of the regulatory limits and generated a comprehensive validation report detailing the exact nature of the non-compliance (Listing 13).

Conforms: False

Results (2):

Constraint Violation in QualifiedValueShapeConstraintComponent:

ITEA4 22003 FireBIM	[CONFIDENTIAL]	45 / 55
D2.3		[DRAFT]



```
Severity: sh:Violation
Source Shape: *The max count sh:PropertyShape*
Focus Node: ex:InvalidCompartment
Result Path: bot:containsZone
Message: A fire compartment may contain a maximum of 4
ResidentialFunctionCaravan or AncillaryFunction.
Constraint Violation in SPARQLConstraintComponent:
Severity: sh:Violation
Source Shape: fbo-
ext:FireCompartmentMaxWagonsAndAncillaryFunctionsShape
Focus Node: ex:InvalidCompartment
Value Node: ex:InvalidCompartment
Source Constraint: *The sh:SPARQLConstraint*
Message: The total usable area of ResidentialFunctionCaravan and
AncillaryFunction within a fire compartment must not exceed 1000 m2.
```

Listing 13 Validation report generated by pySHACL

4.4.4. GraphRAG-Driven Rule Retrieval and Compliance Checking

The final demonstrator integrates the natural language retrieval of regulations with automated SHACL validation using a Graph Retrieval-Augmented Generation (GraphRAG) architecture. In this scenario, the OpenSmartHome model was semantically enriched with fire safety triples, and the regulation graph was populated with SHACL shapes representing Articles 4.50 and 4.51. When a user submitted a natural language query regarding fire compartments and residential functions, the GraphRAG pipeline translated the prompt into a SPARQL query to retrieve the URLs of the corresponding regulations and their associated SHACL shapes.

The pySHACL engine then iteratively evaluated the retrieved shapes against the building data. Because the model was intentionally modified to leave one space outside a fire compartment, it failed the first check (Listing 14) while passing the second (Listing 15). The resulting violation report was subsequently processed by an LLM agent to automatically generate actionable, stakeholder-specific feedback, such as advising the architect to coordinate with fire engineers to resolve spatial boundaries during early design (Listing 16).

SHACL Validation Report for http://www.firebim.org/regulations#NL-RO-OmW-BBL-4-4.2-4.2.8-4.50-1.0_SHACL

Validation Report

Conforms: False

Results (1):

Constraint Violation in MinCountConstraintComponent

(<http://www.w3.org/ns/shacl#MinCountConstraintComponent>):

Severity: sh:Violation

Source Shape: [sh:class fbo-nl:FireCompartment ; sh:message

Literal("An enclosed space must be contained within a fire

compartment.") ; sh:minCount Literal("1", datatype=xsd:integer) ;

sh:path [sh:inversePath bot:containsZone]]

Focus Node: :Room

Result Path: [sh:inversePath bot:containsZone]

Message: An enclosed space must be contained within a fire compartment.

Listing 14 Validation report of article 4.50 member 1

All nodes targeted by http://www.firebim.org/regulations#NL-RO-OmW-BBL-4-4.2-4.2.8-4.51-2.0_SHACL are valid.

Listing 15 Validation report of article 4.51 member 2

As the Architect in the early design phase, your primary duty is to resolve the SHACL violation by ensuring every room is assigned to a FireCompartment to comply with Bbl performance requirements. You must immediately coordinate with the Fire Engineer and Structural Engineer to integrate fire-resistant boundaries into your



architectural layout, as these will dictate both the building's spatial safety and its cost profile. Ensure all fire safety principles are documented for municipal review under the Omgevingswet, as your design serves as the legal foundation for the building permit. By resolving these data gaps now, you protect the project from non-compliance during the eventual handover to the Contractor and Building Control.

Listing 16 Generated stakeholder-specific solution to the validation report (stakeholder: architect, project phase: early design)

4.5. Rule exploration and retrieval

4.5.1. Semantic enrichment and thematic metadata tagging

The regulation graph captures the structure of the document and the digital rules, and based on that structure, the regulations can be queried. However, there is a need to enable rule exploration and retrieval based on terms from the FBO ontology, and not just based on the structure of the document, to enable faster and more dynamic retrieval. This subsection therefore explains how the regulation graph is enriched with thematic metadata that explicitly links individual regulatory statements to concepts from a domain ontology. In the FireBIM implementation, the FireBIM Regulation Ontology is combined with the FireBIM Building Ontology using the `fbo:isAbout` object property to link regulatory statements with relevant domain topics.

The thematic enrichment (Figure 21) is performed at the level of individual regulatory statements. The statements are retrieved from the regulation graph and evaluated against the selected ontology concepts using a zero-shot text-classification approach. The language model determines which topic is most strongly associated with the text. When the resulting similarity score exceeds the defined threshold, the corresponding ontology concept is added to the regulation graph using an `fro:isAbout` relation. This results in triples of the form:

`ex:Statement fro:isAbout fbo:Topic`

where `ex:Statement` represents the regulatory statement containing the original text and `fbo:Topic` identifies the corresponding domain concept.

This approach adds semantic metadata to the regulatory graph without requiring the regulatory text itself to be rewritten or structurally transformed. It also enables retrieval based on hierarchical ontology concepts. For example, a query for `fbo:FireCompartment` can include more specific national or domain-specific subclasses through the ontology hierarchy. This is particularly relevant when regulations from different jurisdictions are represented in a common knowledge model, since a harmonised domain concept can be linked to jurisdiction-specific definitions.

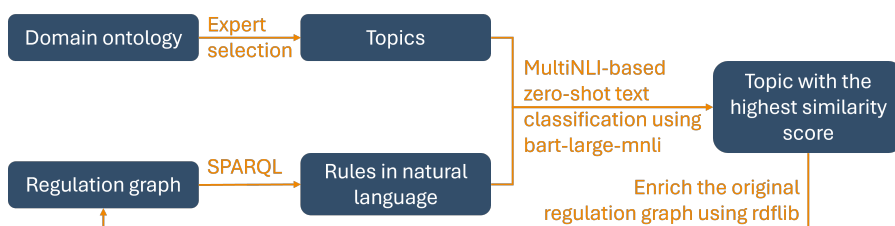


Figure 21 Method for LLM-driven graph enrichment

One limitation of the current approach is that thematic tagging is performed at statement level. A theme explicitly represented in one statement is not automatically propagated to related statements, for example, an exception contained elsewhere in the same article. The authors therefore identify hierarchical propagation of metadata as an area for further development.



4.5.2. GraphRAG architecture for rule retrieval

The semantically enriched regulation graph forms the basis for rule retrieval. SPARQL is the native query language for RDF graphs, however, creating SPARQL queries requires specialised knowledge. Our standpoint is that end-users should not need to be able to write SPARQL queries in order to retrieve the digital regulations from the graph. We therefore introduce a Graph Retrieval Augmented Generation (GraphRAG) architecture in which users write queries in natural language and a Large Language Model enriched with symbolic knowledge representations translates this question into SPARQL queries.

The GraphRAG architecture (Figure 22) consists of three elements: 1. The natural language interface, 2. The LLM-based query generation, and 3. The regulation graph. The LLM receives the user's question (in natural language) together with the semantic context (ontologies and example question/SPARQL sets) to interpret it. The generated SPARQL query is executed directly against the RDF database containing the regulation graph. The resulting regulations, including their original textual content, are then returned to the user. In the implemented approach, the LLM is used to generate the query only. The answer that is returned to the end-user is a direct result of the SPARQL query, and the LLM itself cannot hallucinate the regulations themselves.

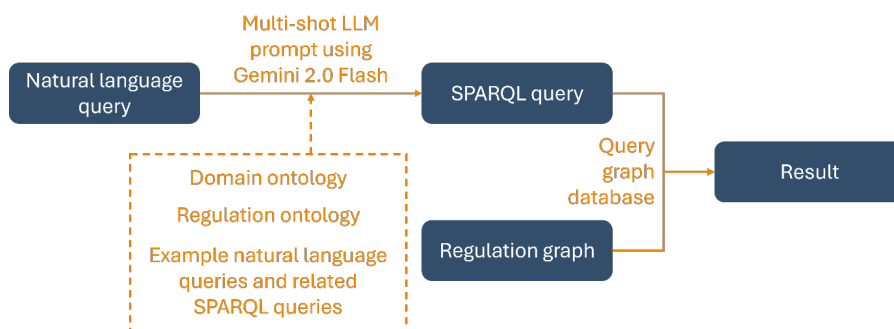


Figure 22 GraphRAG-based dynamic SPARQL generation

The GraphRAG method was implemented as the FireBIM Chatbot, a web-based interface that allows users to retrieve digital building regulations using natural-language questions. The user does not need to create SPARQL queries themselves. Instead, the Large Language Model interprets the question and generates a corresponding SPARQL query, which is executed against the regulatory knowledge graph. The retrieved regulatory statements are then presented to the user. The chatbot can also display the generated SPARQL query alongside the retrieved regulations. This provides transparency into how the natural-language question was interpreted and allows users to inspect the query underlying the retrieval results.

For example, the user can ask a fire-safety-related question in natural language, such as requesting all regulations concerning **fire compartments and residential functions**, including their English text. The chatbot translates this question into a graph query and returns the matching regulatory statements from the FireBIM knowledge graph, including their regulation identifier and original text (Figure 23).

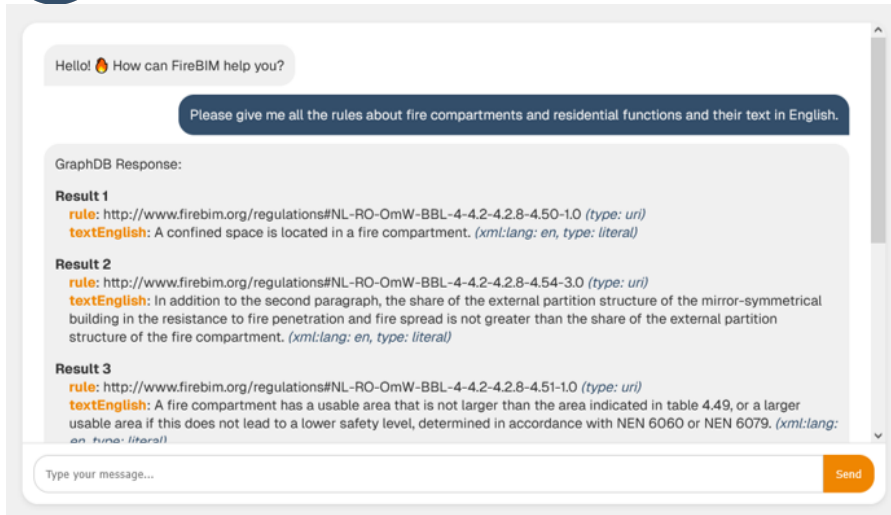


Figure 23 FireBIM Chatbot as a GraphRAG implementation

The chatbot can also combine different building-use concepts, for example asking for the Dutch text of regulations concerning industrial or healthcare functions. The response consists of the relevant regulatory statements and their original Dutch text retrieved from the knowledge graph (Figure 24).

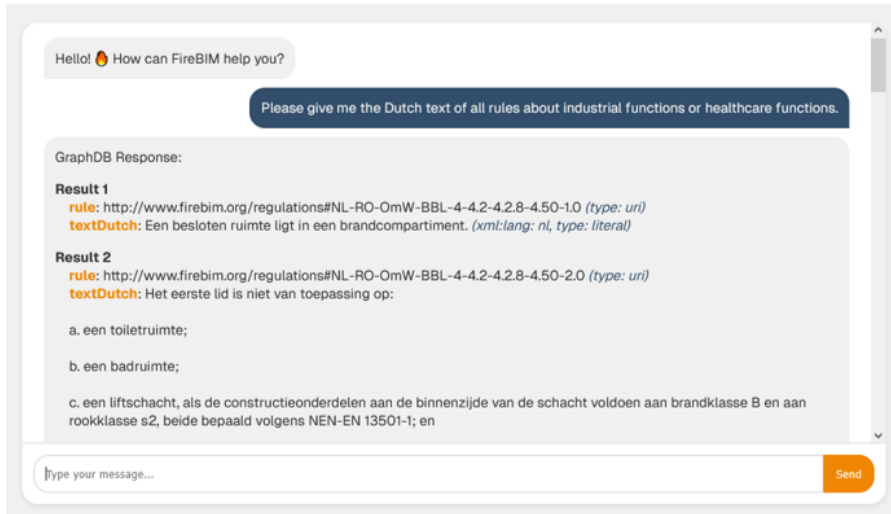


Figure 24 FireBIM Chatbot as a GraphRAG implementation, part 2



5. Conclusion

This deliverable has presented a generic framework for the digitalisation of fire safety regulations. The framework addresses the complete path from regulatory documents in heterogeneous formats to machine-interpretable rules that can be applied to digital building information. The central principle is to separate the different steps in this process and to make each step independently inspectable and, where possible, deterministic.

The work resulted in four main elements: a regulation graph, a rules graph, a Linked Building Data graph, and a compliance report. Together, these elements provide the basis for connecting normative knowledge with building information and executing automated compliance checks.

The regulation graph provides a common representation for regulations from different countries. The FireBIM Regulation Ontology separates the structure of a regulatory document from its country-specific implementation. This proved useful because the input documents differ substantially. Dutch regulations can make use of existing XML structures, whereas other regulations require additional preprocessing or LLM-based interpretation. The resulting regulation graphs nevertheless follow the same conceptual structure. This supports the intended generic architecture, while recognising that the preprocessing of individual regulatory sources cannot be fully standardised.

Several approaches were investigated for interpreting regulatory text. The work shows that there is no single method that is sufficient for all regulatory statements. RASE provides a transparent expert-driven representation, while NLP methods provide a more deterministic way of extracting logical elements from text. LLM-based methods can support the generation of intermediate decision trees and the interpretation of more heterogeneous documents. The neuro-symbolic approach combines these capabilities with explicit semantic and logical representations. In particular, the use of an intermediate First-Order Logic representation separates the interpretation of a regulation from its final implementation in SHACL.

SHACL was selected as the main executable representation because it can express constraints on RDF data and can be combined with SPARQL-based targets and geometric computations. At the same time, the work shows that the conversion from natural language to SHACL is not a purely technical transformation. Regulatory interpretation contains assumptions, exceptions, implicit knowledge, and context. Human validation is therefore retained as an explicit part of the workflow. The intermediate representations, together with the SHACL-to-text and SHACL-to-diagram transformations, provide mechanisms through which fire safety experts can inspect the resulting rules.

The demonstrators show that the resulting rules can be executed against Linked Building Data. IFC models can be converted to RDF, enriched with fire safety information, and combined with geometric calculations. The resulting data can then be checked using SHACL. The demonstrators successfully identified deliberately introduced violations involving fire compartments, MEP elements, spatial containment, cardinality, and area limits. The final demonstrator further showed how natural-language rule retrieval can be connected to SHACL validation through a GraphRAG architecture.

There are also clear limitations. The digitalisation process is not fully automatic. The source regulations remain dependent on their national format and structure, and expert validation is required to establish whether the digital rule correctly represents the source regulation. The current thematic enrichment is also performed at statement level, which means that relations between statements, such as exceptions expressed elsewhere in an article, require further development. More extensive validation across countries, regulations, and rule types is needed before the framework can be considered mature. The work also does not attempt to replace fire safety experts or legal interpretation. Instead, it provides a digital representation that makes regulatory knowledge available for systematic processing and checking.



Within Work Package 2, D2.3 therefore acts as the logical bridge between the domain knowledge established in D2.1 and the formal Linked Building Data structures provided by D2.2, and the software and rulesets developed in D2.4 and D2.5. The FireBIM terminology and ontology provide the semantic basis for interpreting regulations, while the workflows and intermediate representations developed in this deliverable provide a basis for implementation. At the same time, the experiments with different machine-readable representations inform the structure of the rulesets to be developed in D2.5. The framework can therefore be seen as the connection between the semantic foundation of WP2 and its implementation.

The next step is to further implement and test these methods in the FireBIM software environment and to extend the available digital rulesets. The main direction is not to remove the human from the process, but to make the interpretation and validation of regulations more systematic, traceable, and reusable. In this way, the framework provides a basis for moving from individually implemented fire safety checks towards a more generic approach to digital fire safety regulations.



References

- Amor, R., & Dimyadi, J. (2021). The promise of automated compliance checking. *Developments in the built environment*, 5, 100039.
- 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.
- Zhang, J., & El-Gohary, N. M. (2017). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in construction*, 73, 45-57.
- P. Pauwels, D.V. Deursen, R. Verstraeten, A semantic rule checking environment for building performance checking [J], *Autom. Constr.* 20 (5) (2011) 506–518,
- P. Pauwels, W. Terkaj, EXPRESS to OWL for construction industry: towards a recommendable and usable ontology [J], *Autom. Constr.* 63 (2016) 100–133,
- B.T. Zhong, L.Y. Ding, H.B. Luo, Ontology-based semantic modelling of regulation constraint for automated construction quality compliance checking [J], *Autom. Constr.* 28 (58) (2018) 58–70
- Y. Lu, Q.M. Li, Z.P. Zhou, Ontology-based knowledge modelling for automated construction safety checking [J], *Saf. Sci.* 79 (2015) 11–18,
- B. Zhong, C. Gan, H.B. Luo, Ontology-based framework for building environmental monitoring and compliance checking under BIM environment [J], *Saf. Sci.* 141 (2018) 127–142
- Fitkau, I., & Hartmann, T. (2024). An ontology-based approach of automatic compliance checking for structural fire safety requirements. *Advanced engineering informatics*, 59, 102314.
- Ismail, A. S., Ali, K. N., Iahad, N. A., Kassem, M. A., & Al-Ashwal, N. T. (2023). BIM-based automated code compliance checking system in Malaysian fire safety regulations: a user-friendly approach. *Buildings*, 13(6), 1404.
- Chen, Y., & Jiang, H. (2024). Optimizing automated compliance checking with ontology-enhanced natural language processing: Case in the fire safety domain. *Journal of Environmental Management*, 371, 123320.
- Wang, Y., Liu, Y., Cai, H., Wang, J., & Zhou, X. (2023). An automated fire code compliance checking jointly using building information models and natural language processing. *Fire*, 6(9), 358.
- Goh, W., Jang, J., Park, S. I., Choi, B. H., Lim, H., & Zi, G. (2024). Automated compliance checking system for structural design codes in a BIM environment. *KSCE Journal of Civil Engineering*, 28(10), 4175-4189.
- Peng, J., & Liu, X. (2023). Automated code compliance checking research based on BIM and knowledge graph. *Scientific Reports*, 13(1), 7065.
- Eastman, C., Eastman, C. M., Teicholz, P. et al. (2011). *BIM Handbook: A Guide to Building Information Modeling for Owner, Manager, Designer, Engineers and Contractors* (p. 243). Hoboken, NJ: John Wiley & Sons.
- Sacks, R., Radosavljevic, M., & Barak, R. (2010). Requirements for building information modeling based lean production management systems for construction. *Automation in construction*, 19(5), 641-655.
- Henttinen (2024) - Regulatory Information Requirements, GUID: 10bb5d17-0215-437b-b9ab-4677cc0f5ad9, Regulatory Domain 2022-2024, based on national requirements from 'England and Wales', 'Scotland', 'Finland', 'Estonia' and 'Singapore'.
- Jos (2023) - Model checking methods for BIM: The state of the art analysis



Dimyadi, J., Clifton, C., Spearpoint, M., & Amor, R. (2014, November). Computer-aided compliance audit to support performance-based fire engineering design. In *Proceedings of 10th International Conference on Performance-Based Codes and Fire Safety Design Methods*.

Lee, Y. C., Eastman, C. M., & Solihin, W. (2018). Logic for ensuring the data exchange integrity of building information models. *Automation in Construction*, 85, 249-262.

Lee, Y. C., Eastman, C. M., & Solihin, W. (2021). Rules and validation processes for interoperable BIM data exchange. *Journal of Computational Design and Engineering*, 8(1), 97-114.

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

Fahad, M., Bus, N., & Fies, B. (2018). Semantic BIM reasoner for the verification of IFC Models. In *eWork and eBusiness in Architecture, Engineering and Construction* (pp. 361-368). Boca Raton, FL, USA: CRC Press.

Luttun, J., & Krijnen, T. (2020, July). An approach for data extraction, validation and correction using geometrical algorithms and model view definitions on building models. In *International Conference on Computing in Civil and Building Engineering* (pp. 529-543). Cham: Springer International Publishing.

Guo, D., Onstein, E., & La Rosa, A. D. (2021). A semantic approach for automated rule compliance checking in construction industry. *IEEE Access*, 9, 129648-129660.

Zhang, R., & El-Gohary, N. (2022). Building information modeling, natural language processing, and artificial intelligence for automated compliance checking. In *Research companion to building information modeling* (pp. 248-267). Edward Elgar Publishing.

Chen, N., Lin, X., Jiang, H., & An, Y. (2024). Automated building information modeling compliance check through a large language model combined with deep learning and ontology. *Buildings*, 14(7), 1983.

Zhang, Z., Ma, L., & Nisbet, N. (2023). Unpacking ambiguity in building requirements to support automated compliance checking. *Journal of Management in Engineering*, 39(5), 04023033.

Nawari, N. O. (2019). A generalized adaptive framework (GAF) for automating code compliance checking. *Buildings*, 9(4), 86.

Lee, H., Lee, J. K., Park, S., & Kim, I. (2016). Translating building legislation into a computer-executable format for evaluating building permit requirements. *Automation in construction*, 71, 49-61.

Yang, F., Zhang, J., Chen, Y., & Debs, L. (2022). A new schema of logic representation and reasoning for automated building code compliance checking. *Proceedings of the GPEA Polytechnic Summit 2022: Session Papers*.

Hjelseth, E. (2012). Converting performance based regulations into computable rules in BIM based model checking software. *eWork and eBusiness in Architecture, Engineering and Construction: ECPPM*, 461-469.

Hjelseth, E. & Nisbet, N. (2010). Exploring semantic based model checking.

de Mendonça, E. A., & Ferreira, S. L. (2024). Use case for automated code compliance checking of accessibility rules in bim models using the ids standard. In *Proceedings of the 41st International Conference of CIB W78*.

Beach, T. H., Rezgui, Y., Li, H., & Kasim, T. (2015). A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert systems with applications*, 42(12), 5219-5231.

Nisbet, N., Zhang, Z., & Ma, L. (2023, July). Automated generation of SPARQL queries from semantic mark-up. European Council on Computing in Construction (EC3).

Dimyadi, J., Clifton, C., Spearpoint, M., & Amor, R. (2014). Regulatory knowledge encoding guidelines for automated compliance audit of building engineering design. In *Computing in Civil and Building Engineering (2014)* (pp. 536-543).



Garrett Jr, J. H., Palmer, M. E., & Demir, S. (2014). Delivering the infrastructure for digital building regulations. *Journal of Computing in Civil Engineering*, 28(2), 167-169.

Fuchs, S., Fauth, J., & Amor, R. (2024). Developing formal representations of building regulations through a parallel drafting process.

Zhang, J. (2017). A logic-based representation and tree-based visualization method for building regulatory requirements. *Visualization in Engineering*, 5(1), 2.

Nisbet, N., Ma, L., & Aksenova, G. (2022). Presentations of rase knowledge mark-up. In *Proceedings of the European Conference on Computing in Construction* (pp. 184-190). University of Turin.

Dimyadi, J., Governatori, G., & Amor, R. (2017, July). Evaluating legaldocml and legalruleml as a standard for sharing normative information in the aec/fm domain. In *Proceedings of the Joint Conference on Computing in Construction (JC3)* (Vol. 1, pp. 637-644). Heraklion, Greece: Heriot-Watt University, Edinburgh, UK.

Lee, J. K. (2011). Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program.

Solihin, W. (2015). A simplified BIM data representation using a relational database schema for an efficient rule checking system and its associated rule checking language.

Stepien, M., Vonthron, A., & König, M. (2023, January 1). OpenBIMRL - An open format for code compliance checking of complex functional requirements from a regulation to the BIM model. *EG-ICE 2023 Conference Papers*. Published. EG-ICE International Conference on Intelligent Computing in Engineering, London. <https://www.ucl.ac.uk/bartlett/sites/bartlett/files/5342.pdf>

Mazairac, W., & Beetz, J. (2013). BIMQL—An open query language for building information models. *Advanced Engineering Informatics*, 27(4), 444-456.

Moult, D., & Krijnen, T. (2020). Compliance checking on building models with the Gherkin language and Continuous Integration. In *EG-ICE 2020 Workshop on Intelligent Computing in Engineering, Proceedings* (pp. 294-303).

Khemlani, L. CORENET e-PlanCheck: Singapore's Automated Code Checking System. 2005. Available online: <http://www.aecbytes.com/buildingthefuture/CORENETePlanCheck.htm> (accessed on 25 March 2023).

BCA. CORENET—About Us. 2021. Available online: <https://www.corenet.gov.sg/general/about-us.aspx> (accessed on 2 March 2023).

Ding, L., Drogemuller, R., Rosenman, M., Marchant, D., & Gero, J. (2006). Automating code checking for building designs-DesignCheck. *Clients driving construction innovation: Moving ideas into practice*, 113-126.

Dimyadi, J., & Amor, R. (2013). Automated building code compliance checking—where is it at. *Proceedings of CIB WBC*, 6(1), 172-185.

Nawari, N. O. (2018). Building information modeling: Automated code checking and compliance processes. CRC Press.

Smith, R. L. (1991). EXPOSURE: An expert system fire code. *Fire technology*, 27(2), 145-159.

GSA. BIM Guide 02 Spatial Program Validation; U.S.G.S. Administration, Ed.; U.S. General Services Administration: Washington, DC, USA, 2020.

Invicara, BIM Assure Deliver Complete, Reliable BIM Data. 2020. Available online: <https://invicara.com/technology/bimassure/> (accessed on 13 December 2022)

Daima. Daima Streamlines the Plan-Checking Process. 2020. Available online: <https://daima.squarespace.com/> (accessed on 18 January 2023).



Fischer, S., Schranz, C., Urban, H., & Pfeiffer, D. (2023). Automation of escape route analysis for BIM-based building code checking. *Automation in Construction*, 156, 105092.

Preidel, C., & Borrmann, A. (2018). BIM-based code compliance checking. In *Building information modeling: Technology foundations and industry practice* (pp. 367-381). Cham: Springer International Publishing.

Shih, S. Y., Sher, W., & Giggins, H. (2013). Assessment of the Building Code of Australia to inform the development of BIM-enabled code checking systems.

Lee, Y. C., Eastman, C. M., & Lee, J. K. (2015). Validations for ensuring the interoperability of data exchange of a building information model. *Automation in Construction*, 58, 176-195.

Greenwood, D., Lockley, S., Malsane, S., & Matthews, J. (2010, September). Automated compliance checking using building information models. In *RICS COBRA 2010 Conference*. RICS.

Kincelova, K., Botton, C., Blanchet, P., & Dagenais, C. (2019). BIM-based code compliance checking for fire safety in timber buildings: A comparison of existing tools.

Werbrouck, J., Hagedorn, P., Donkers, A. J. A., & Fauth, J. (2026). Semantic Web Technologies for Building Permitting. In J. Fauth, D. Guler, R. Lavikka, F. Noardo, P. N. Gade, & S. M. Ventura (editors), *Digital Transformation of Building Permitting* (blz. 153-168). Routledge Taylor & Francis Group. <https://doi.org/10.1201/9781003650287-12>

Hagedorn, P., Fauth, J., Zentgraf, S., Seif, S., König, M., & Brilakis, I. (2025). OntoBPR: An ontology-based framework for performing building permit reviews using standardized information containers. *Advanced Engineering Informatics*, 66, 103369.

Donkers, A. J. A., & Petrova, E. (2026). A Neuro-Symbolic AI Pipeline for Generating SHACL-Based Digital Building Regulations. In *Proceedings of the 2026 European Conference on Computing in Construction*.

Donkers, A. J. A., & Petrova, E. (2024). Converting Fire Safety Regulations to SHACL Shapes Using Natural Language Processing. In E. Vakaj, S. Iranmanesh, R. Stamartina, N. Mihindukulasooriya, S. Tiwari, F. Ortiz-Rodríguez, & R. Mcgranaghan (editors), *NLP4KGC 2024 : Natural Language Processing for Knowledge Graph Creation 2024: Proceedings of the 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation co-located with 20th International Conference on Semantic Systems (SEMANTICS 2024)* (CEUR Workshop Proceedings; Vol. 3874). CEUR-WS.org. <https://ceur-ws.org/Vol-3874/>

Shi, J. W. L., Solihin, W., & Yeoh, J. K. (2025). Fine-tuning a large language model for automated code compliance of building regulations. *Advanced Engineering Informatics*, 68, 103676.

Madireddy, S., Gao, L., Din, Z. U., Kim, K., Senouci, A., Han, Z., & Zhang, Y. (2025). Large language model-driven code compliance checking in building information modeling. *Electronics*, 14(11), 2146.

Iversen, O., & Huang, L. (2026). Leveraging large language models for BIM-based automated compliance checking. *Automation in Construction*, 182, 106707.

Hettiarachchi, H., Dridi, A., Gaber, M. M., Parsafard, P., Bocaneala, N., Breitenfelder, K., ... & Vakaj, E. (2025). CODE-ACCORD: A Corpus of building regulatory data for rule generation towards automatic compliance checking. *Scientific data*, 12(1), 170.

Zentgraf, S., Hagedorn, P., & König, M. (2026). A BIM-based framework for automated building code extraction and compliance checking. *Advanced Engineering Informatics*, 74, 104735.

Lee, J. H., Ji, S. Y., & Ostwald, M. J. (2026). Automated compliance checking across the building lifecycle: Systematic and semantic review integrating PRISMA and deep search. *Automation in Construction*, 185, 106859.

Mirhosseini, N., Shojaei, D., & Sabri, S. (2026). A systematic review of methods for interpreting building code regulations in automated compliance systems. *Building Research & Information*, 1-24.