Advanced Engineering Informatics 51 (2022) 101449

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Multi-ontology fusion and rule development to facilitate automated code compliance checking using BIM and rule-based reasoning

Liu Jiang [a,b], Jianyong Shi [a,b,*], Chaoyu Wang [a]

[a] Department of Civil Engineering, Shanghai Jiao Tong University, Shanghai, China
[b] Shanghai Key Laboratory for Digital Maintenance of Buildings and Infrastructure, Shanghai, China

ARTICLE INFO

Keywords:
Automated compliance checking
Ontology
Rule-based reasoning
Building information modeling
Chinese building codes

ABSTRACT

Code compliance checking plays a critical role that identifies substandard designs according to regulatory documents and promises the accuracy of the designs before construction. However, the traditional code compliance checking process relies heavily on human work. To help the users better understand the checking process, this study proposes a gray-box checking technique and a BIM-based (Building Information Modeling) automated code compliance checking methodology that leverages ontology. The proposed approach contains a code ontology, a designed model ontology, a merged ontology, a code compliance checking ontology, a set of mapping rules, and a set of checking rules. During the checking process, the ontologies provide knowledge bases, and the rules provide necessary logic. A five-step roadmap is proposed for code ontology development for domain experts. For the time being, pre-processing is applied to create the designed model ontology to achieve time saving. Next, an ontology mapping procedure between the code and the designed model ontology is executed to obtain the merged ontology. In the ontology mapping procedure, the mapping rules aim to mitigate the semantic ambiguity between design information and regulatory information and enrich building information's semantics. Subsequently, rule-based reasoning is applied based on the checking rules and the merged ontology for checking reports generation. Finally, according to Chinese building codes, an automated code compliance checking platform is implemented for real construction projects to validate the proposed methodology.

1. Introduction

Code compliance checking of building design is one of the critical issues in the AEC (Architecture, Engineering, and Construction) industry. A traditional manual checking process, an iterative process based on 2D drawings, is error-prone and low efficient. Thus, automated code compliance checking has caught continuous interest in the AEC domain with the advent of building information modeling (BIM). The automated code compliance checking process can be divided into four phases [1]: rule interpretation phase, building model preparation phase, rule execution phase, and rule check-reporting phase. The rule interpretation phase, which translates the regulatory codes into a machine-readable language, is the starting point and the core task of the automated compliance checking process. One practical way is to translate the regulatory codes based on hard-coded program methods. For example, the COnstruction and Real Estate NETwork (CORENET) platform embed the codes into the C++ programs [2], Express Data Manager (EDM) platform published by Jotne EPM Technology [3] translates the codes into EXPRESS data modeling language, the Solibri Model Checker (SMC) platform interprets the codes into a set of predefined rule templates. However, such a checking process is an inflexible method and demands considerable programing skills [42]. In addition, extensive work is required for system maintenance when the regulatory codes are revised or updated [4]. In addition, due to legal restrictions, transparency of the checking process is needed. For the users and domain experts involved in the checking process, knowing how the rules are executed is important for verifying the availability and responsibility of the checking results.

The basic issue of the building model preparation phase is the information access, retrieval, and model derivation. Generally, the data standard of the building model, IFC (Industry Foundation Classes), has been regarded as a data source for an automated code compliance checking system [5–7]. The building information could be obtained by parsing the IFC files. However, a mapping between IFC and regulatory codes is necessary due to two reasons. Firstly, because IFC is designed to

* Corresponding author at: Shanghai Jiao Tong University, No. 800 Dongchuan Road, Minhang District, Shanghai 200240, China.
E-mail address: shijy@sjtu.edu.cn (J. Shi).

https://doi.org/10.1016/j.aei.2021.101449
Received 6 May 2021; Received in revised form 22 October 2021; Accepted 1 November 2021
Available online 12 November 2021

describe the information of a building model during the overall life cycle, it is recommended to create the subset on account of checking the scope of the regulatory codes before the rule execution phase [8]. Secondly, there exists some semantic ambiguity between the entities of the IFC standard and the concepts of regulatory codes. In particular, this ambiguity comes from two aspects. On the other hand, multiple pairs of synonym terms may exist in regulation and design documents as a result of nonstandard and inconsistent terminology usage, such as "floor," "story," "elevator shaft," and "elevator space." Additionally, most design documents contain abbreviations for convenience. Therefore, it is necessary to consider a formal description of the concepts; moreover, IFC lacks implicit semantics. For instance, the entity "IfcSpace" could represent either kitchen or a bedroom, but these concepts differ in regulatory information. In addition, before performing the testing procedure, some code-specific abstract concepts such as "basement," "stair railing," and "kitchen door" should be clearly stated. Afterward, the prepared rules are applied to the designed model, and the checking reports are presented with results such as "pass," "fail," or "warning," and "unknown" [1] in the last two phases. It is noted that the correctness and completeness of the building information is the basic prerequisite for high-quality and trustable checking results. As such, the semantic enrichment of the building model is required [9].

This paper proposes an automated code compliance checking method developed based on ontology and semantic web technologies. The advantage of establishing the automated code compliance checking within the semantic web environment is that it allows for describing the compliance checking process in the same data model or language [10]. Accordingly, the design and regulatory information could be matched in the same description model (i.e. ontology), which has provided a suitable environment to mitigate ambiguity. Some of the key recent investigations for ontology-based code compliance checking are summarized in Table 1. Compared to these researches, this paper has three contributions; firstly, the code ontology designed primarily for code compliance checking is used to describe regulatory information. A five-step road map is proposed for code ontology development for domain experts and engineers. Secondly, a designed model ontology created by modifying the local structure of ifcOWL [41] is utilized to describe design information. By comparison, the proposed designed model ontology has more direct relationships between IFC entities and requires less file processing time. At last, this paper establishes a system framework based on Apache Jena [11] to integrate all the functional modules, and the proposed framework is practical, feasible, and extensible. The proposed platform could meet the following particular requirements: (1) the checking process is understandable to domain experts; (2) the semantic gaps between IFC entities and the design code concepts could be well addressed; (3) a sub-model for code compliance checking of the building model could be created automatically; and (4) the information completeness of the building model could be prechecked.

Table 1
Summary of recent studies of ontology-based code compliance checking.

Reference	Method description	Limitations
[13–14,17]	Proposed regulatory ontology for code compliance checking and utilized both OWL axioms and semantic web rules to describe regulatory documents.	No methods to deal with information retrieval from building models and to deal with ambiguous information. No promise for the completeness of the building model.
[10,12]	Retrieved building information from IFC-related files and utilized semantic web rules or RDF graphs to describe checking rules.	No methods to deal with ambiguous information. No promise for the completeness of the building model.
[15,16]	Proposed a framework for describing regulatory information, design information, and rules for code compliance checking.	Rules are SPARQL rules. No promise for the completeness of the building model.

The paper is organized as follows. Section 2 reviews recent literature on automated, rule-based checking and identifies the research gap. Section 3 proposes a framework for the automatic compliance checking platform based on BIM and ontology and presents brief introductions to five main functional modules of the platform. Section 4 shows the feasibility and efficiency of the proposed framework using an example. Section 5 concludes the study and elaborates on future work.

2. Background and review of related studies

Eastman et al. [1] has given some definitions of automated, rule-based checking. First of all, automated rule checking was defined as "*software that does not modify a building design, but rather assesses a design on the basis of the configuration of objects, their relations or attributes.*" Then the rule-based system refers to the system that "*applies rules, constraints or conditions to a proposed design, with results such as 'pass,' 'fail,' or 'warning', or 'unknown,' for cases where the needed data is incomplete or missing.*" In view of these definitions, the rule-based checking system contains three critical components: a schema, a set of instances, and a set of rules [10]. Moreover, Beach et al. [18] pointed out that a rule-based semantic approach should consider all the related semantics, including the semantics of the domain itself (i.e., the schema), the semantics of the data formats (i.e., the set of instances), and the semantics of the regulations (i.e., the set of rules). Therefore, this section provides a brief review of the efforts to achieve these underpinning semantics—specifically, (1) regulatory ontology, which supports the semantics of the domain itself, (2) ifcOWL, which supports the semantics of the data formats, and (3) rule languages which supports the semantics of the regulations.

2.1. Regulatory ontology for code compliance checking

An ontology is a formal representation of knowledge in a specific domain by a set of concepts, properties, and relations [19]. Due to the advantages of providing a shared understanding of knowledge among domains and harmonizing the interpretation of various data models [20], the ontology has been introduced for improving domain knowledge sharing and reuse [21–22], heterogeneous data integration [23], business management [24,25] and so on. To describe the ontology model in a machine-processable way, an ontology description language, the Web Ontology Language (OWL), is developed and recommended by the World Wide Web Consortium (W3C). Specifically, OWL ontology could be divided into three sub-languages: OWL-Lite, OWL-DL, and OWL-Full. OWL-lite is the simplest language used for describing simple classes hierarchy and simple constraints, while OWL-Full is the most expressive sub-language but lacks the ability to perform automatic reasoning. As an expansion of OWL-Lite, OWL-DL is a better choice for representing the language of ontology. Meanwhile, the ontology editing tool, Protégé, could be used for ontology prototype development.

In general, the semantics of the domain itself is provided by a designed regulatory ontology. The regulatory ontology aims at organizing domain knowledge and disambiguating the semantics of professional terms. Zhong et al. [13] designed a quality inspection and evaluation ontology (CQIEOntology) to model the generic concepts and relations in the construction quality compliance checking. Representing in OWL language, a set of OWL axioms was also added in the CQIEOntology to support the checking process. Lu et al. [17] proposed a construction safety checking ontology (CSCOntology) including five primary classes (i.e. "Line of Work," "Task," "Precursor," "Hazard," and "Solution") to express safety checking concepts. Some researchers developed the ontology based on the combination of regulatory ontology and additional domain ontologies, such as the building environmental monitoring ontology [15] and the utility product ontology [16]. In this research, the code ontology is developed based on the

regulatory documents for providing the semantics of the compliance checking domains.

2.2. Ontology heterogeneity between ifcOWL and regulatory ontology

The semantics of building data could be obtained with the transformation from IFC data to RDF (Resource Description Framework) data [12,16]. The key issue of the data transformation is the mapping between the EXPRESS schema and Web Ontology Language (OWL), a machine-interpretable language to support conceptualizing concepts and creating ontologies based on the semantic web. The first conversion mapped from EXPRESS to OWL was presented by Schevers and Drogemuller [26]. From then on, the following researches have focused on the ontology-based representation of IFC EXPRESS schema. Recently, Terkaj [27] and Pauwels [28] have proposed an EXPRESS-to-OWL converter of IFC data model and the ifcOWL (an OWL representation of the IFC schema) has become a standard in buildingSMART International. Based on the ifcOWL and a designed regulatory ontology, automated code compliance checking could be implemented for acoustic performance checking [29] and building environment checking [15].

However, since different organizations or research institutions developed, ontology heterogeneity appears between ifcOWL and the regulatory ontologies when executing the automated compliance checking process. The heterogeneity is mainly caused by the *mismatches of different conceptualizations of the same domain* [30], namely the conceptualization mismatches. Firstly, the coverage of the regulatory ontology and ifcOWL differs. The ifcOWL is considered as an OWL version of the full IFC schema [10]; the regulatory ontologies, which could be regarded as experts in a certain domain, focus on the information related to their concerns. Secondly, the terminology used in regulatory ontology and ifcOWL differs. For example, the regulatory term "wall" is defined as "IfcWallStandardCase" in ifcOWL, and the regulatory term "exterior wall" is defined by the entity "IfcWallStandardCase" and the property "isExternal" in ifcOWL. Thirdly, the granularity of regulatory ontology and ifcOWL differs. The relations between concepts in ifcOWL are much more complex for application in practice [31]. Finally, the description perspectives of ifcOWL and regulatory ontology are different. Because IFC is designed for building data exchange and sharing, the structure of ifcOWL is different from that of regulatory ontology developed from regulatory documents designed for human reading.

Therefore, the ontology mediation between regulatory and ifcOWL is considered a priority to support the automated code compliance checking platform. The ontology mediation involves the schema level mediation, which could address the differences in coverage and granularity, and the instances level mediation, which could address the differences in terminology and descriptions. The schema level mediation aims to reconstruct the model structure of ifcOWL and then apply the modular ifcOWL on a specific purpose. The concept of modular ifcOWL was firstly proposed by Terkaj & Pauwels [32] and has been adopted in several types of research, such as the Building Topology Ontology [31] and ifcOWL-DfMA Ontology [33]. The instances level mediation is intended to map the concepts with similar semantics between ifcOWL and regulatory ontology by a set of mapping rules. As such, from the aspect of schema level mediation, a designed model ontology is developed by modifying the local structure of ifcOWL in this paper. And a set of mapping rules are designed for the instances level mediation between the designed model ontology and the code ontology.

2.3. Rule language for regulatory documents

One of the most challenging issues in automated code compliance checking is rule interpretation. Many current research has focused on white-box approaches that transform the regulatory codes into computer-readable rule languages. Recently, based on the semantic web technologies, the SPARQL queries [14–16] or dedicate semantic web rule languages, such as semantic web rule language [13,17] and the N3Logic language [10], are implemented for code compliance checking to provide the semantics of the regulations. These languages are also friendly to human reading and are easily encoded manually by domain experts, such as the Building Environment Rule and Analysis (BERA) language [34], KBimLogic, and KBimCode [35], Drools rule language [36], and legal mark-up languages [4]. On combining the RASE (requirement, applicability, selection, exception) representation mechanism, Macit İlal & Günaydın [37] proposed a four-level hybrid representation model for building codes.

In general, based on the regulatory ontology, ifcOWL, and rule language, the prototype of a fully semantic environment for automated compliance checking has unfolded. In this research, ontology development is the initial step of semantic web-based compliance checking. The ontology development includes regulatory ontology development and design ontology development. As the second step of semantic web-based code compliance checking, ontology mapping is considered an ontology mediation approach to reconcile regulatory and design ontology heterogeneity. Finally, rule-based reasoning is executed. Besides, a set of mapping rules and checking rules are designed for ontology mapping and rule-based checking.

3. The proposed methodology

As illustrated in Fig. 1, this paper proposes an automated code compliance checking methodology based on ontology and BIM. The suggested method includes four key ontologies and two sets of rules. A code ontology, a designed model ontology, a merged ontology, and code compliance checking ontology are the four ontologies where the code ontology was designed to represent knowledge from building codes and serves as a TBox. The code ontology also specifies what types of information are utilized and how they are organized throughout the rule-based checking process. The designed model ontology, often known as the ABox, is used to describe knowledge about building models, including design information containing a set of instances asserting facts based on the concepts defined in the TBox. In the following reasoning procedure, the merged ontology, created by mapping the designed model ontology with the code ontology, serves as a combination of TBox and ABox. After reasoning, the code compliance checking ontology is produced, which includes the checking results. The mapping and checking rules are the two sets of rules that provide the necessary logic. Furthermore, two key techniques, namely ifcOWL structural reorganization and ontology mapping, are used to address the aforementioned ontology heterogeneities for coverage, granularity, description, and terminology and accomplish semantic enrichment of the building model. Furthermore, a completeness check of the merged ontology is performed to verify that all essential information is acquired prior to reasoning. The proposed methodology seeks to equalize ontology, data, and rule development; the details are given in the rest of this section.

3.1. Code ontology development

The code ontology acts as a knowledge base of the platform. A practical method to create an ontology is an iterative method proposed by Stanford University, including seven steps [38]: domain determination, existing ontologies reuse, important terms enumeration, classes and classes hierarchy definition, properties definition, and facts definition instances creation. Based on this seven-step method, a five-step development roadmap is designed to create the code ontology in this research, as shown in Fig. 2.

The initial step is to target the knowledge area of the ontology. The code ontology is developed to answer two basic questions: which building elements will be checked and what properties need to be checked. Since developed based on regulatory documents, domain experts are required to select terminologies from the clauses sentences. These terminologies are categorized into class taxonomies and property

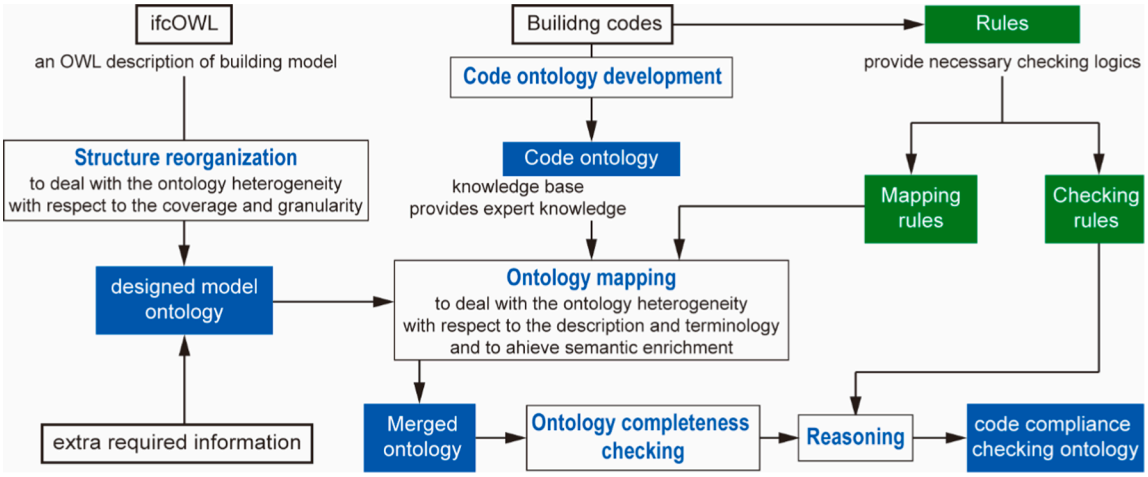


Fig. 1. Automated code compliance checking Framework based on BIM and ontology.

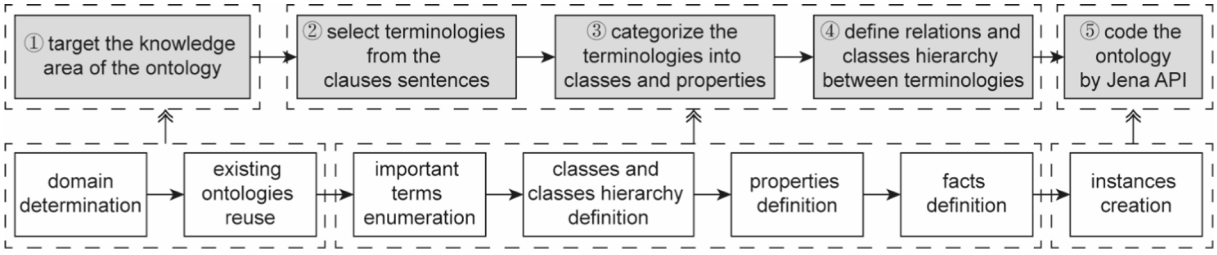


Fig. 2. Five-step roadmap for code ontology development.

taxonomies. Next, the relations and classes hierarchy between these terminologies are defined. Finally, the ontology is coded by Apache Jena Ontology API. Additionally, the proposed code ontology focuses on the building elements, the attributes and geometry property of the building elements. But it has potential to be combined with the LegalRuleML [45] to express more semantics of the regulatory documents in further researches.

3.2. Designed model ontology development

The designed model ontology is developed to represent knowledge of building models and related design information. The knowledge is derived from the BIM models, which are considered one of the most important knowledge resources, calculation files, project documents, etc. This paper focuses on the knowledge capture from IFC data format, widely used ISO standard in the AEC industry, and is seen as the source of BIM model; but the knowledge completion from other files or user inputs for the designed model ontology is also supported.

Since the IFC is designed for data exchange for the whole information of the life-cycle of the building, the information of whole IFC model is redundant for automated code compliance checking task that is domain-restricted according to different checking scenario. Thus, the structure reorganization of ifcOWL is applied to derive a subset of the building model. First, the input IFC file is converted into ifcOWL by the IFCtoRDF converter [39]. Then, the designed model ontology was developed based on the ifcOWL by modifying the ifcOWL structure locally. The modification includes element structure reorganization and the properties extraction, and the entities have the prefix "De" to avoid repetition and ambiguity in the designed model ontology.

The element structure reorganization process is to define straightforward relations between elements which are related through six IFC entities (i.e. IfcRelAggregates, IfcRelSpaceBoundary, IfcRelContainedInSpatialStructure, IfcRelVoidsElement, IfcRelFillsElement, and IfcRelConnectPathElement), as shown in Fig. 3. Three new ObjectProperty, namely "De:has", "De:connect", and "De:relatedTo" are defined in the designed model ontology. The "De:has" means that one element is a part of another element, such as IfcWallStandardCase, IfcSpace, IfcStairFlight, and IfcStair. Both "De:connect" and the "De:relatedTo" mean that two elements are connected, but the former implies that the two elements are connected in order.

In the properties extraction process, the property information may be extracted through three IFC entities in this paper: IfcRelDefinesByType, IfcRelDefinesByProperties, and IfcGloballyUniqueId. The three entities describe the basic properties (Pset), type properties (Typeset), and GUID property. Each pair of property names and values will be stored in the designed ontology as DatatypeProperty and data of the corresponding elements, as shown in Fig. 4.

The designed model ontology could yield two benefits. On the one hand, the size of the ontology file could be reduced so that the file processing could be faster in the following module. The comparisons of file size and reading time between ifcOWL and designed model ontology are listed in Table. 2. Because the designed model ontology is considered a specific view of the ifcOWL, the ontology file size may be decreased by at least 85 %. The converting time of the converting process (i.e., the element structure reorganization process and the properties extraction process) depends on the number of IFC entities. For example, Model①, Model②, and Model④ are architecture-designed, while Model③ is a designed water supply and drainage system model. Thus, it takes more time for ontology converting of Model③ because more IFC entities are required to describe the water pipes. Moreover, it is noted that the converting process costs much more time than reading ifcOWL, but the converted designed model ontology could be used for subsequent tasks so that the file reading time could be saved when reading the ontology file repeatedly. For instance, if the ifcOWL ontology of Model④ is used twice in the code compliance checking task, it will cost $235 \times 2 = 470$ seconds. However, it will only cost $270 + 1 \times 2 = 272$ seconds after

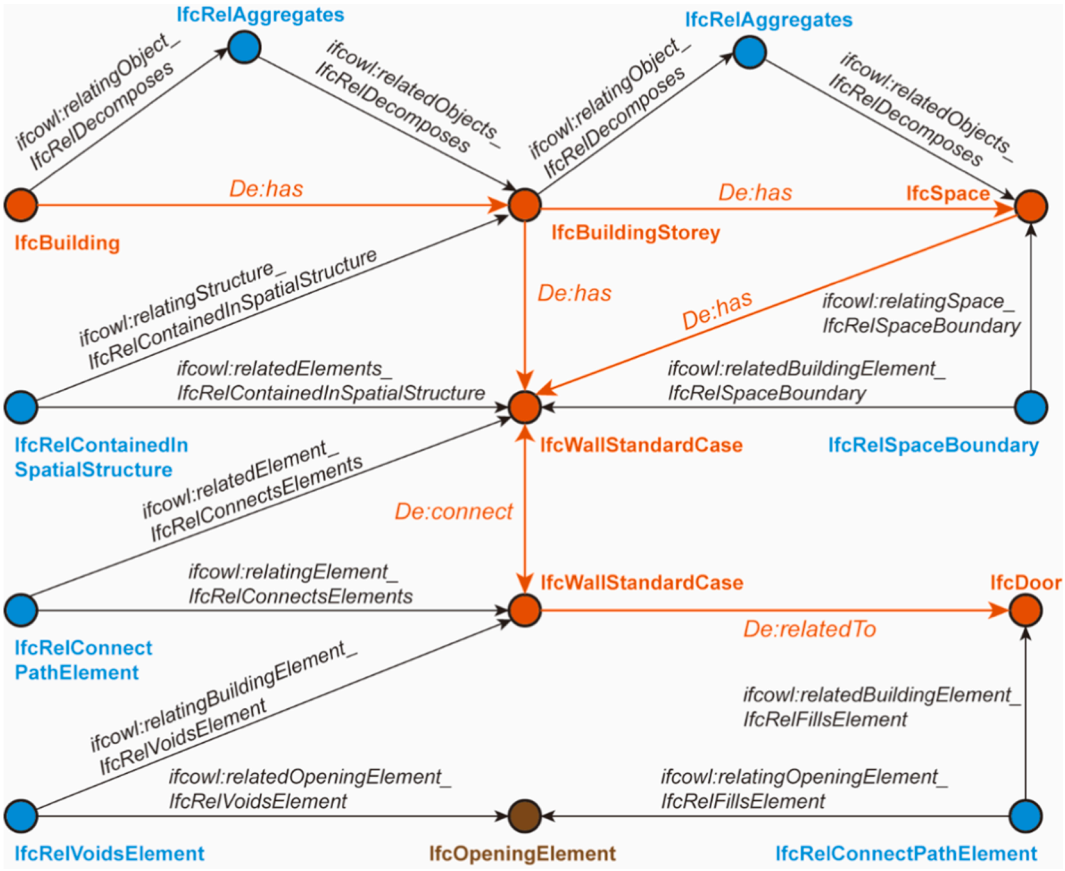


Fig. 3. Examples to illustrate the element structure reorganization process.

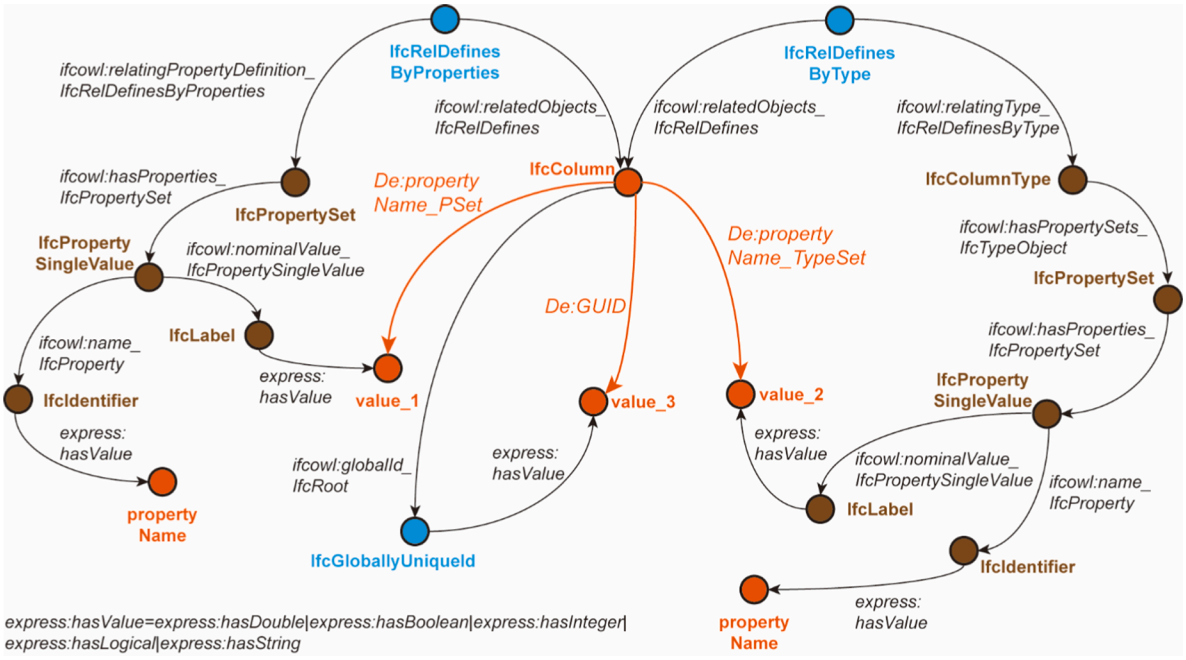


Fig. 4. Examples to illustrate the properties extraction process.

converting the ifcOWL ontology into the designed model of Model④.

On the other hand, the development of mapping rules could be simplified because new terms in the designed model ontology could be used. Since the mapping rules are manually developed by domain experts, illustrated in Section 3.3, the simplified development process could alleviate manual errors. For example, as shown in Fig. 5, the mapping rule of a flat beam between designed model ontology and code ontology is much more simplified than that between ifcOWL and code ontology. A flat beam is defined as a beam whose width is greater than or equal to its depth. The depth and the width of a beam are defined by a

Table 2
Comparisons between ifcOWL and designed model ontology.

Model	File size (MB)			File reading time (s)		Converting time (s)	
	ifcOWL	Designed model ontology	Storage reduce ratio (%)	ifcOWL	Designed model ontology	including ifcOWL reading	Without ifcOWL reading
Model①	43.8	6.17	85.91	4	0.6	11	7
Model②	415	22.9	94.48	28	1	56	28
Model③	1076	59	94.52	70	3	253	183
Model④	2970	19.6	99.34	235	1	270	35

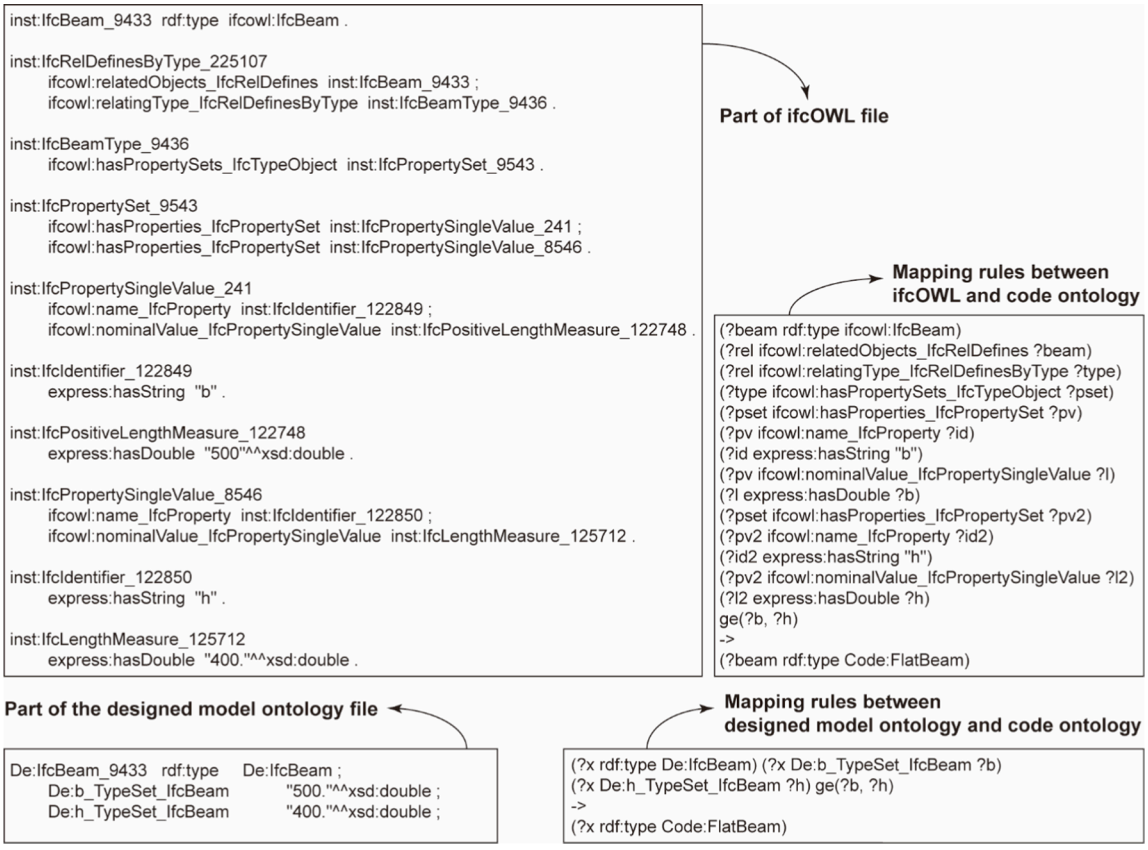


Fig. 5. Mapping rules of flat beam.

sequence of IFC entities in the ifcOWL ontology, and it requires fifteen terms in the mapping rules. On the contrary, the beam's section height and width are defined by a "De:h_TypeSet_IfcBeam" property and a "De:b_TypeSet_IfcBeam" property, respectively. Thus, only four terms are required for creating mapping rules between designed model ontology and code ontology.

3.3. Mapping rules development and ontology mapping

The rules provide complex logic of automated code compliance checking. Even though the most recently developed standard ontology description language OWL could support concepts description and reasoning, a limited range of reasoning tasks could be figured out [40]. Therefore, additional semantic rule language is necessary to meet the complex logical representation of the checking process. This research selected the Jena rule as the rule language because it could support forward-chaining, backward-chaining, and hybrid-reasoning inference. It also provides a set of built-in functions for handling mathematical operations, string dealing, and value comparison. To achieve complex mapping, the SPQRAL-query and external programming are necessary. The logic for automated code compliance checking involves two parts, i.e., mapping rules and checking rules. Domain experts are invited to participate in these two parts. As mentioned in section 3.2.1, the new relations or terms (e.g. "De:has," "De:connect," and "De:relateTo") derived in the designed model ontology could be used in the mapping rules. The mapping rules are used to map the code ontology and the designed model ontology to obtain a merged ontology. The ontology mapping is proposed to mediate the ontology heterogeneities on terminologies and descriptions between the two ontologies and enrich the merged ontology semantics in this paper. Besides, it is noted that user input or pre-tagging of data is necessary to provide adequate information. However, ontology mapping aims at obtaining as much information and semantics as possible through reasoning or some appropriate algorithms while consuming as lesser work of data tagging as possible.

To deal with different tasks, the mapping rules could be categorized into relations deriving rules, property mapping rules, and entity mapping rules. Because of avoiding repetition and ambiguity, the entities of the designed model ontology have the prefix "De," and the entities in the code ontology have the prefix "Code." The relations deriving rules are utilized for creating new relationships between entities. For example, the rule "(?r1 rdf:type Code:Room)(?r2 rdf:type Code:Room)(?r1 De:has ?d)(?r2 De:has ?d)(?d rdf:type Code:Door)->(?r1 Code:accessTo ?r2)(?r2 Code:accessTo ?r1)" indicates that the room (?r1) and (?r2) have the same door (?d) and thus are accessible to each other.

The property mapping rules include terminology mapping rules, inherent mapping rules, and calculation mapping rules. The terminology mapping rules are mainly used for dealing with the ambiguities of description. To make the concepts in the checking rules developed based on the regulatory codes clearer to users, all the property terminologies are asserted equivalent with the corresponding concepts in the regulatory codes. The inherent mapping rules aim at creating new properties for the entities by inheriting from relationships. For instance, the rule "(?x rdf:type Code:Railing)(?level rdf:type De:IfcBuildingStorey)(?level De:has ?x)(?level De:storeyNumber ?v)->(?x Code:locates ?v)" is used to assign the location property for the railings. The calculation mapping rules emphasize creating new properties by a set of processing. On the one hand, if the new property only involves one entity, a Jena rule could be used to figure out the generation process. For example, the rule "(?x rdf:type Code:beam)(?x Code:height ?h)(?x Code:span ?s) quotient(?h, ?s, ?ratio)->(?x Code:depthSpanRatio ?ratio)" is used to obtain the depth-span ratio for a beam. On the other hand, if the creation of the new property involves multiple entities, external functions are required. For instance, as shown in Fig. 6, the generation of window-to-floor-area ratio property restricted by the lightning windows of a room contains three steps. Firstly, a SPARQL query is used to collect the whole lighting window area for the room. Then, each pair of the selected results is added to the ontology. Finally, a Jena rule is utilized to calculate the window-to-floor-area ratio for the room. Here the lightening window refer to the window that lets the natural lights into the room. The window-to-floor-area ratio is calculated by the window area divided by the floor area.

The entity mapping rules could be further classified into terminology mapping, decompose mapping, and aggregation mapping. The terminology mapping is utilized to address the differences in terminologies or descriptions, such as a rule 3-1 for mapping the entity "IfcSpace" in the designed model ontology with the entity "Room" in the code ontology and rule 3-2 for mapping the entity "IfcDoor" and the entity "Door," as shown in Table. 3. The differences in terminologies also include some synonyms definitions. For example, the concept "living room" and the concept "storey" are equivalent to the concept "sitting room" and the concept "floor" separately.

The decompose mapping focus on fractioning and refining the entities based on their own properties or relations with other entities. For instance, the IFC entity "IfcSpace" could represent either "Kitchen" or "Bedroom", but these two descriptions are distinct in regulatory files. So, the decompose mapping rules (rule 3-3 and rule 3-4 in Table. 3) based on the "Name" properties are used to enrich the semantics of "IfcSpace" entities. According to the properties calculation, rules 3-5 and 3-6 are two decomposing mapping rules to identify the basement and semi-basement for the entity "Room." As shown in Fig. 7, if the difference between the elevation of the outdoor ground and the elevation of the indoor floor is larger than half of the room's height, the room is identified as a basement. Similarly, if the difference between the two elevations varies from one-third to half of the room height, the room will be identified as a semi-basement. According to the relations with other

Table 3
Examples to illustrate different types of entity mapping.

Rule type		Rule No.	Example rules
Terminology mapping		3-1	(?x rdf:type De:IfcSpace) -> (?x rdf:type Code:Room)
		3-2	(?x rdf:type De:IfcDoor) -> (?x rdf:type Code:Door)
Decompose mapping	Based on their own properties	3-3	(?x rdf:type De:IfcSpace) (?x De:Name_Pset "Kitchen") -> (?x rdf:type Code:Kitchen)
		3-4	(?x rdf:type De:IfcSpace) (?x De:Name_Pset "Bedroom") -> (?x rdf:type Code:Bedroom)
		3-5	(?x rdf:type Code:Room) (?x Code:ElevationOfOutdoorGround ?ov) (?x Code:ElevationAtTheBottomOfRoom ?rv) (?x De:has ?slab) (?slab rdf:type Code:Slab) (?slab Code:height ?sh) sum(?rv,?sh,?v) (?x Code:clearHeight ?h) quotient(?h, 2, ?mh) sum(?v,?mh,?result) lessThan(?result,?ov) -> (?x rdf:type Code:Basement)
		3-6	(?x rdf:type Code:Room) (?x Code:ElevationOfOutdoorGround ?ov) (?x Code:ElevationAtTheBottomOfRoom ?rv) (?x De:has ?slab) (?slab rdf:type Code:Slab) (?slab Code:height ?sh) sum(?rv,?sh,?v) (?x Code:clearHeight ?h) quotient(?h, 2, ?mh) sum(?v,?mh,?r1) greaterThan(?r1,?ov) quotient(?h, 3, ?th) sum(?v,?th,?r2) lessThan(?r2,?ov) -> (?x rdf:type Code:SemiBasement)
	Based on relations with other entities	3-7	(?x rdf:type Code:Railing)(?s rdf:type Code:Stair)(?s De:has ?x) ->(?x rdf:type Code:StairRailing)
		3-8	(?x rdf:type Code:Door) (?r rdf:type Code:Kitchen) (?r De:has ?x) -> (?x rdf:type De:KitchenDoor)
Aggregation mapping		Rely on external algorithm to achieve complex relational calculation and logical processing.	

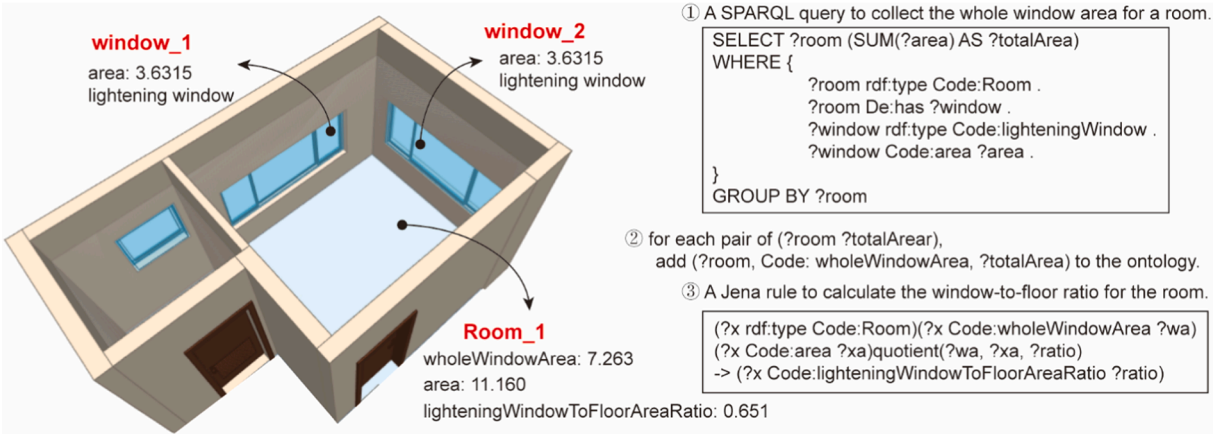


Fig. 6. Algorithm to obtain the window-to-floor ratio for a room.

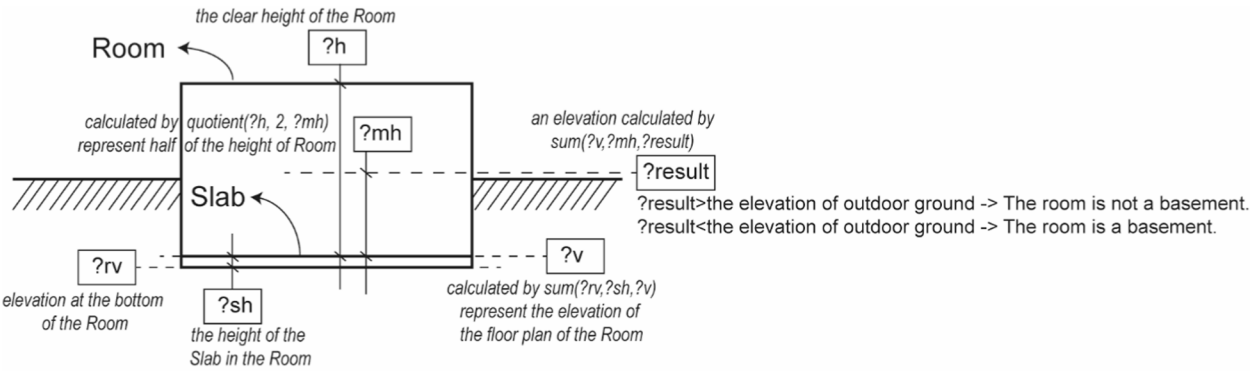


Fig. 7. Sketch map of Rule 3-5.

entities, the decompose mapping rules 3-7 and 3-8 are applied for stair railing and kitchen door detection.

The aggregation mapping emphasizes spatial enrichment, such as mapping the dwelling and the fire zone, consisting of rooms or spaces. Because the aggregation mapping contains complex relational calculation and logical processing in most instances, external algorithms could be developed as a built-in module of the mapping rules to capture some features or as a functional module to accomplish mapping directly.

3.4. Completeness checking of ontology

Subsequently, a completeness checking process is executed for the merged ontology to ensure all necessary information for code compliance checking is obtained. Completeness checking is the process of checking the existence of particular statements in the merged ontology. For example, when executing code compliance checking for the clause "The indoor clear height of bedroom should not be lower than 2.40 m," all "Bedroom" entities should have a property of "IndoorHeight." Then, the completeness checking is to ensure that the statement "*(Bedroom, Code:IndoorHeight, value)*" is valid for every "Bedroom" entity.

3.5. Checking rules and reasoning

The checking rules are interpreted from code clauses by domain experts. Generally, the rules could be divided into four classes according to [8]: *class 1-rules that require a single or a small number of explicit data; class 2-rules that require simple derived attribute values; class 3-rules that require extended data structure; class 4-rules that require a "proof of solution."* Some examples of the checking rules are listed in Table. 4. The rules checking the property for one entity are assigned as class-1, such as the rule 4-1 and rule 4-2. Rule 4-1 indicates that the distance of baluster (?v) of the stair railings (?x) should not be less than 110 mm. Rule 4-2 is developed from the clause "The window-to-floor-area ratio of the lightening window of the bedroom should not be less than 1/7." Though the property "lighteningWindowToFloorAreaRatio" should be derived from a group of attribute values, it could be figured out by a simple Jena rule because the property derivation has been accomplished by the ontology mapping process, as shown in Fig. 6. However, this clause also indicates that the entity "bedroom" should have at least one entity, "lighteningWindow." As such, rule 4-2 is a combination checking process.

The checking rules could also be executed for checking the non-spatial or non-numeric relations between two entities. For instance, rules 4-3 are used to check relative functional rooms for a dwelling. If the user wants to check the number of the entities, a SPARQL query works more efficiently for the numeric relations checking. Taking rule 4-4 as an example, the storey (?storey) with no or only one egress will be selected and regarded as a substandard entity in the ontology. Besides, the entities that need to be checked with corresponding solutions are heightened with "Warning" in the ontology, such as rule 4-5 that is used to remind users to check the protective facilities of the external windows. This kind of rules could be assigned to class-1, class-2 or class-3 rules because they still rely on checking with single explicit data, simple derived attribute values or extended property.

Checking rules for geometric properties or spatial relations between two entities, as regarded as a combination of class-2 and class-3 rules, necessitate the use of external functions and some pre-filled data. External functions may be developed in other programming languages and then packaged as self-defined built-in Jena rules functions. For example, rule 4-6 is used to check the spatial relations between a toilet and a living room. The toilets which are located on the top of a living room are against the code. In this rule, external functions are necessary for obtaining the geometry of the room and for the built-in function "doublication(?geo1, ?geo2, ?t)", which return a result (?t) as a Boolean. Similarly, an external function should be executed for the built-in function "distance(?geo1, ?geo2, ?d)" to calculate the distance between two geometric objects in rule 4-7, ensuring the layout of two egresses is reasonable. However, due to the complex mathematical processing for geometry, these rules remain to be realized in further work. This paper mainly focuses on the other four types of checking rules.

Finally, a rule-based reasoning process is executed based on the merged ontology, the checking rules, and the Jena reasoning engine. Then a code compliance checking ontology is then created on the combination of the merged ontology and the checking results stored as a list of new facts. Secondly, SPARQL queries (a kind of query language designed to support semantic searching for RDF data) in Fig. 8 is used to retrieve the checking results and the corresponding globally unique identifier (GUID). The checking results are classified into three groups: (1) a "Pass" group which contains the standard-compliant entities; (2) a "Warning" group whose entities are required to be re-checked manually; and (3) a "Failed" group which contains the substandard building components.

4. Case study implementation

4.1. Implementation environment

To inspect the feasibility of the proposed methodology, an automated code compliance checking platform is established for actual engineering projects. The cooperation among the mentioned five subsystems and the interaction between the user and the automated code compliance checking platform are illustrated in Fig. 9. Users can upload the building model for code compliance checking, retrieve the checking logics, obtain pre-check results, and download the checking reports. The proposed system framework contains five main functional subsystems: code ontology development, designed model ontology converter, rules management, ontology mapping, and completeness

Table 4
Examples of checking rules.

Rule types	Rule class	Rule No.	Example rules
Check the property for one entity	Class-1	4-1	(?x rdf:type Code:stairRailing) (?x Code: distanceOfBaulaster ?v) lessThan (?v,110) -> (?x Code:Failed "ClauseGB50096_6_3_2")
	Class-2	4-2	① (?x rdf:type Code:bedroom) (?x Code: lighteningWindowToFloorAreaRatio ? ratio)ge(?ratio, 0.14286) -> (?x Code:Pass "ClauseGB50096_7_1_5") ② (?x rdf:type Code:bedroom) noValue(?x Code:Pass "ClauseGB50096_7_1_5") -> (?x Code:Failed "ClauseGB50096_7_1_5")
Check non-spatial/non-numeric relations for two entities	Class-1	4-3	① (?x rdf:type Code:dwelling) (?x Code:contains ?bedroom)(?bedroom rdf:type Code:bedroom) (?x Code:contains ?livingroom)(?livingroom rdf:type Code:livingRoom) (?x Code:contains ?kitchen)(?kitchen rdf:type Code:kitchen) (?x Code:contains ?bathroom)(?bathroom rdf:type Code:bathroom) -> (?x Code:Pass "ClauseGB50096_5_1_1") ② (?x rdf:type Code:dwelling) noValue(?x Code:Pass "ClauseGB50096_5_1_1") -> (?x Code:Failed "ClauseGB50096_5_1_1")
Check the numeric relations between two entities	Class-1	4-4	SELECT ?storey WHERE { ?storey rdf:type Code:storey. ?storey De:has ?egress. ?egress rdf:type Code:egress. ?egress Code:count ?num .} GROUP BY ?storey HAVING (SUM(?num) less than 2)
Check the solutions	Class-1/ Class-2/ Class-3	4-5	(?x rdf:type Code:externalWindow) (?x Code:heightOfWindowSill ?v) lessThan(?v, 900) (?sr rdf:type Code:staircase)(?sr De:has ?x) (?br rdf:type Code:balcony) noValue(?br De:has ?x) -> (?x Code:Warning "ClauseGB50096_6_1_1")
Check the geometric properties or spatial relations between two entities	Class-2/ Class-3	4-6	(?r1 rdf:type Code:toilet) (?r1 Code: hasGeometry ?geo1) (?r2 rdf:type Code: livingRoom) (?r2 Code:hasGeometry ? geo2) doublication(?geo1, ?geo2, ?t) equal (?t, "true"^^xsd:boolean) -> (?r1 Code: Failed "ClauseGB50096_5_4_4")
	Class-2/ Class-3	4-7	(?storey rdf:type Code:storey) (?storey De: has ?e1) (?storey De:has ?e2) notEqual(?e1,?e2) (?e1 rdf:type Code:egress)(?e1 Code: hasGeometry ?geo1) (?e2 rdf:type Code:egress)(?e2 Code: hasGeometry ?geo2) distance(?geo1, ?geo2, ?d)lessThan(?d,5) -> (?e1 Code:Failed "ClauseGB50096_6_2_4") (?e2 Code:Failed "ClauseGB50096_6_2_4")

checking, and checking reports generation. During the checking process, the code ontology development and the rules management are the two static subsystems. These two subsystems rely on the efforts of developers and domain experts to accomplish the code ontology development, mapping rules preparation, and checking rules interpretation before the code compliance checking process. The other three subsystems handle the business procedure of the code compliance checking automatically. The designed model ontology converter module transforms the input IFC files into the designed model ontology by modifying the local structure of ifcOWL. Then, the ontology mapping and completeness checking subsystem executes mapping rules between the designed model ontology and code ontology to obtain the merged ontology used for model completeness checking. If the merged ontology passes the model completeness checking, the rule-based reasoning engine will be activated for checking reports generation. Otherwise, model modification is required.

The platform is implemented based on Apache Jena [11]. The Jena RDF API and Jena Ontology API are two fundamental tools for supporting ontology reading, writing, and operation. The uploaded building model is converted to ifcOWL ontology by the IFCtoRDF converter [39] firstly. After that, Jena ARQ is utilized for converting the ifcOWL ontology into the designed model ontology. Then, Jena Inference API supports all the reasoning events, including ontology mapping, completeness checking, and rule-based reasoning.

4.2. Implementation preparation

The preparation of the case study contains two aspects, building model preparation, and static modules preparation. In this research, a practical 11-storey residential building model, which is designed based on Autodesk Revit 2020, is used as a test case. SADI Architectural Design Institute provides the tested model. Based on the tested model, some necessary data tags such as the functions of each room and the types of each window have been added to the model for implementation. Part of the revised building mode is delineated in Fig. 10, and this part of the building model could be provided for an external party if required.

Thirty clauses involving parameters checking, relations checking, geometric checking, and spatial checking are selected from Designed Code for Residential Building (GB50096) for code compliance checking. Based on the selected clauses, domain experts are invited to develop a code ontology, mapping rules, and checking rules. Part of the code ontology is shown in Fig. 11. Firstly, 63 terminologies are collected from the clauses and divided into class entities (including 47 terminologies) and data properties (including 16 terminologies). The data properties are categorized into several subsets, such as position, geometric, and environmental properties. Secondly, each clause is translated into mapping rules and checking rules based on the selected terminologies. It is noted that the preparation of the rules is an iterative procedure. Domain experts could retrieve the knowledge graph of the rules to ensure that the checking rules work on accurate logic, which plays a key role in making the checking process more readable and understandable to human.

During the preparation of the mapping rules and checking rules, these rules could be converted into a knowledge graph to present corresponding logic and make the checking process understandable to the users. For instance, the rules for determining a room's lighting widow-to-floor-area ratio (as stated in Fig. 6 and Table 4) may be represented

Pass group

```
SELECT ?e ?guid ?code WHERE {
        ?e Code:Pass ?code .
        ?e De:GUID ?guid .
}
```

Warning group

```
SELECT ?e ?guid ?code WHERE {
        ?e Code:Warning ?code .
        ?e De:GUID ?guid .
}
```

Failed group

```
SELECT ?e ?guid ?code WHERE {
        ?e Code:Failed ?code .
        ?e De:GUID ?guid .
}
```

Fig. 8. SPARQL queries for retrieving checking results.

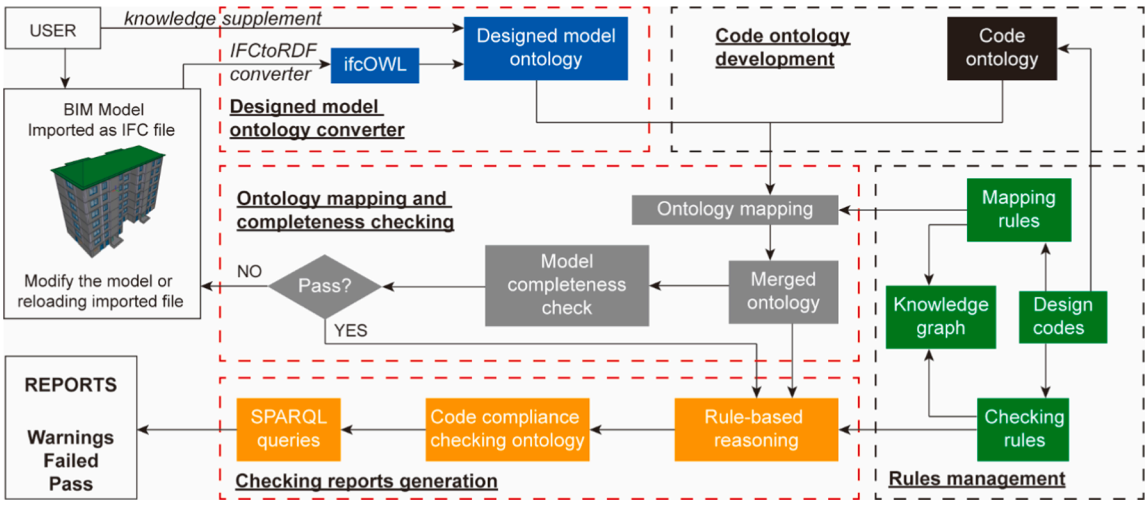


Fig. 9. System framework of the automated code compliance checking platform.

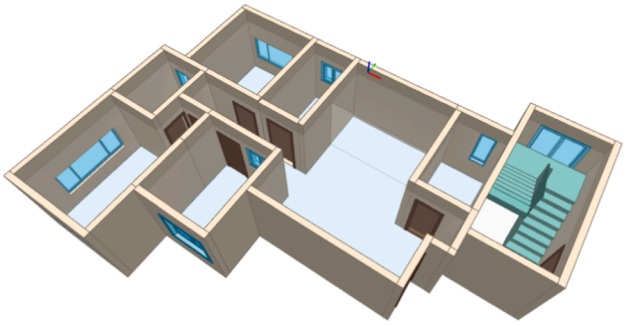


Fig.10. Part of the building model used in the case study.

as a knowledge graph, as shown in Fig. 12. The notations of the knowledge graph are derived from those in the Conceptual Graph proposed by W. Solihin and C. Eastman [44]. A rectangle represents a concept in the knowledge graph, and a circle represents a property connected to its corresponding concept by a solid line. The mapping rules are represented by the line starting with a solid dot and ending with an arrow. The arrow points to the mapping direction, and the mapping logics are specified with a dotted box indicating a Jena rule while a double solid line box indicating an operation based on SPARQL queries. Similarly, the checking rules are represented with a line starting with a solid numeric circle and ending with an arrow. A new notation, namely the numeric line, is proposed to represent the procedural dealing. The lines marked with the same number describe the reasoning process of the same checking rule.

4.3. Implementation results

The remaining three modules produce the implementation results: the model ontology converter module, the ontology mapping and completeness checking module, and the check report generation module. In the designed model converter module, the input building model is converted into ifcOWL ontology and the designed model ontology in succession. Because the IFC file is verbose and redundant, it is recommended to apply the IFC redundancy removal method described in [43]. The comparisons between the original IFC file and the compressed IFC file are listed in Table. 5. As can be seen in the table, the converting time is reduced by 27.7 s. Meanwhile, 53.90% and 40% decrease in the file sizes of the IFC file and the ifcOWL ontology, respectively, after redundancy removal. From the perspective of user interaction, file

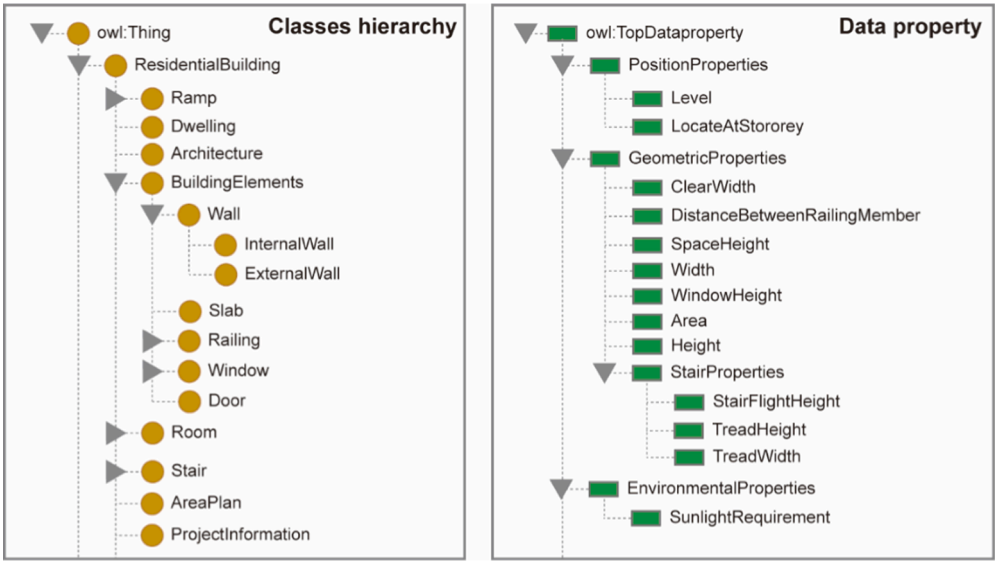


Fig. 11. Part of the code ontology.

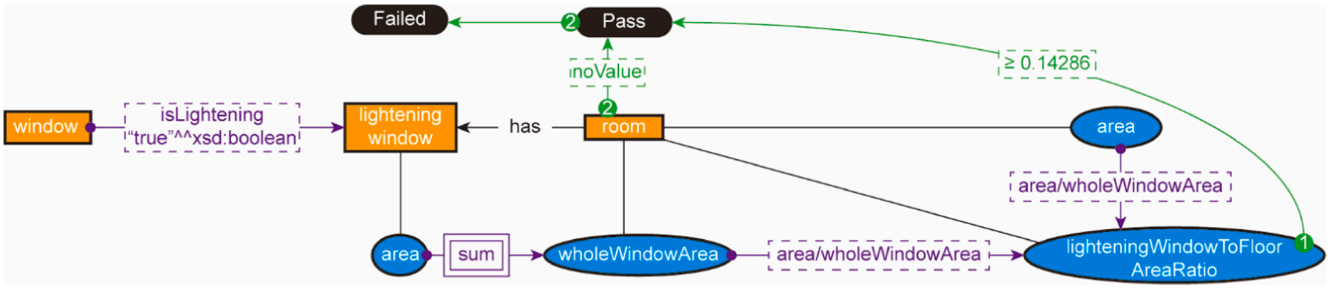


Fig. 12. An example knowledge graph presenting the mapping rule and checking rule.

Table 5
Comparisons between the original IFC file and the compressed IFC file.

File	File size (MB)			Converting time (s)	
	IFC file	ifcOWL ontology	Designed model ontology	From IFC file to ifcOWL ontology	From ifcOWL ontology to designed model ontology
Original IFC file	66.6	415	22.9	41.53	62
Compressed IFC file	30.7	249	22.9	26.83	49

uploading time may be reduced as a result of the file storage decrease.

The input IFC file is first converted into an ifcOWL file and then extracted as the designed model ontology by reorganizing the local structure as mentioned in section 3.2. As shown in Fig. 13, a new relation "has" is created between the room and the corresponding building components through the IFC entity "IfcRelSpaceBoundary" in the designed model ontology. Similarly, the relation "relatedTo" between a wall and a door is derived from the IFC entity "IfcRelVoidsElement" and "IfcRelFillsElement". Besides, based on the transitive relations between a room to a door connected by a wall, generating a "has" relation between a room and a door is now much easier.

Meanwhile, the user could also add some extra information for the designed model ontology. This information focuses much on global information of the building model, such as the building function, the seismic design information of the whole building, the structure type, and many others. All these knowledge supplementations are collected as a DatatypeProperty in the designed model ontology.

A merged ontology could be created by combining the code ontology and the designed model ontology based on the developed ontology mapping rules. Parts of the merged ontology are shown in Fig. 13 and Fig. 14. Because of having the same door, the space entity "IfcSpace_1868" and the space entity "IfcSpace_1333" are accessible by each other in the merged ontology. It is noted that the IfcSpace entities should be tagged with the specific room types (i.e., "kitchen" and "living room"). Nevertheless, due to the inherent advantage of ontology that describes data in a relational graph, a number of semantic enrichments could be figured out by formulating related mapping rules. For example,

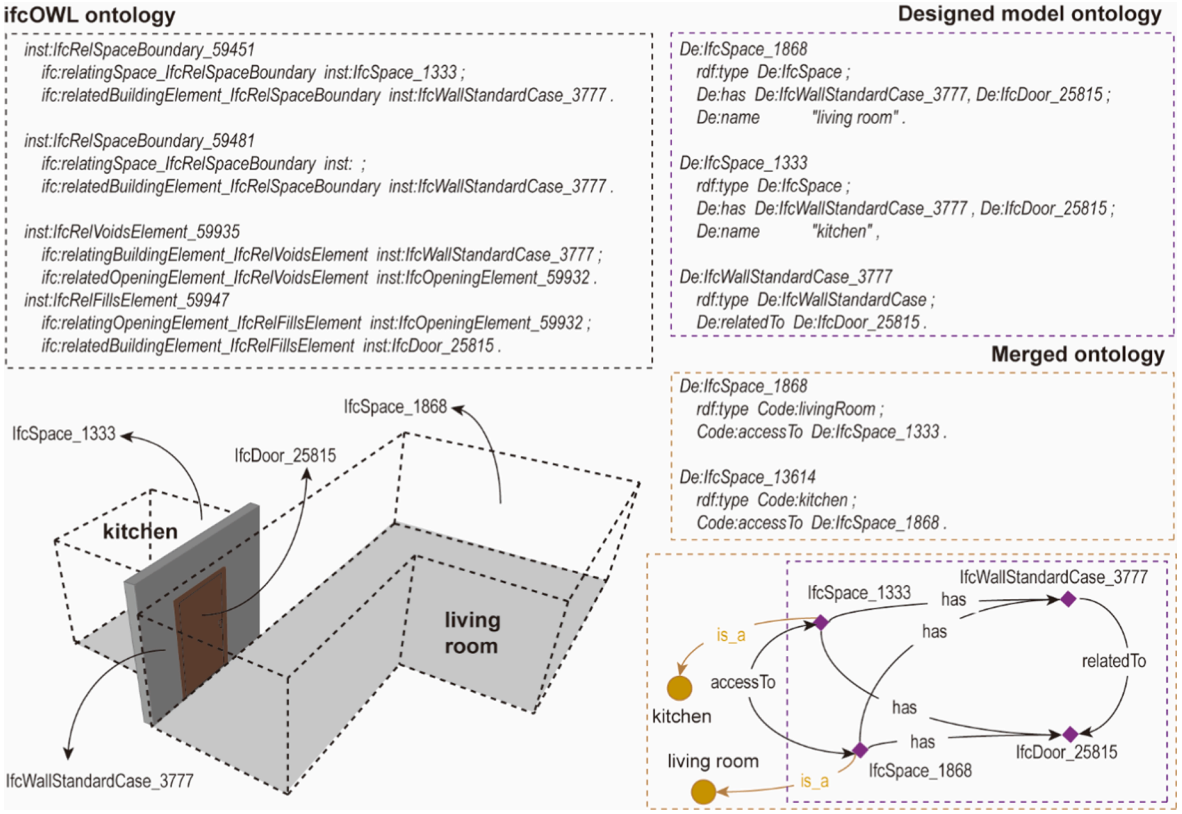


Fig. 13. Part of the designed model ontology and the merged ontology.

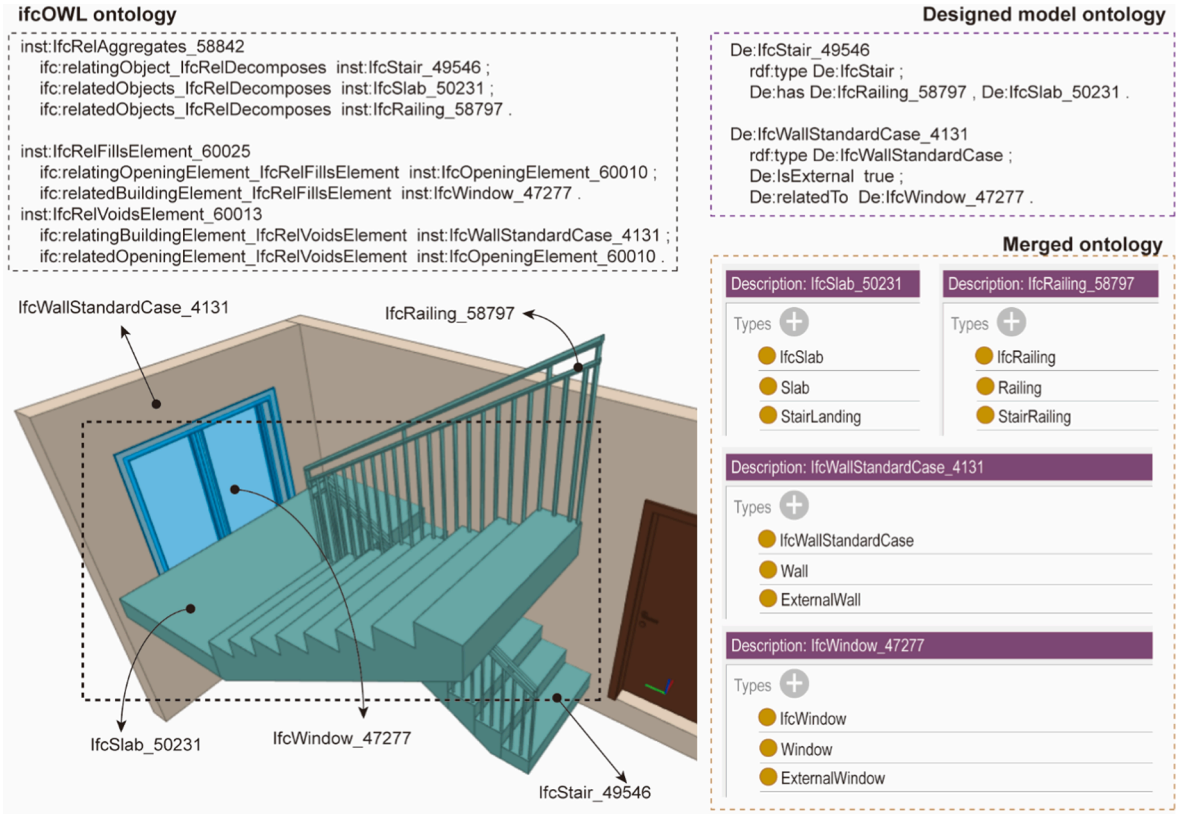


Fig. 14. Semantic enrichment in the merged ontology.

as shown in Fig. 14, the "IfcRailing_58797" is identified as a stair railing and the "IfcSlab_50231" is identified as a stair landing because of being an accessory of the assembling stair "IfcStair_49546." Besides, the "IfcWindow_47277" is identified as an external window because the attached wall "IfcWallStandardCase_4131" is an external wall. The mapping rules listed in Table 6 are utilized to achieve these goals.

Besides, the rules 6-2 and 6-3 in Table 5 are selected for an additional comparison between a normal code compliance checking process and the proposed code compliance checking process in this paper; the results are shown in Table 7. Obviously, the executing time of the whole process is reduced to 191.114 s. Even though the proposed process requires an additional 62 s for file pre-processing, the business procedure executes considerably faster than the normal process since the relationships between IFC entities are simplified significantly in the designed model ontology. Additionally, the more mapping rules that are executed, the more time is anticipated to be saved by the proposed code compliance checking process.

Subsequently, the completeness checking process on the merged model is executed. Users could preview the checking reports online or download the checking reports if the designed model is passed in the model completeness checking process. Otherwise, users will obtain the pre-check results, which could be used as a model modification guide. The checking reports include the GUID of the building component, the error types, and the checking details. Fig. 15 illustrates parts of the checking results for the lightening window-to-floor area ratio.

Table 6
Examples of mapping rules to achieve semantic enrichments.

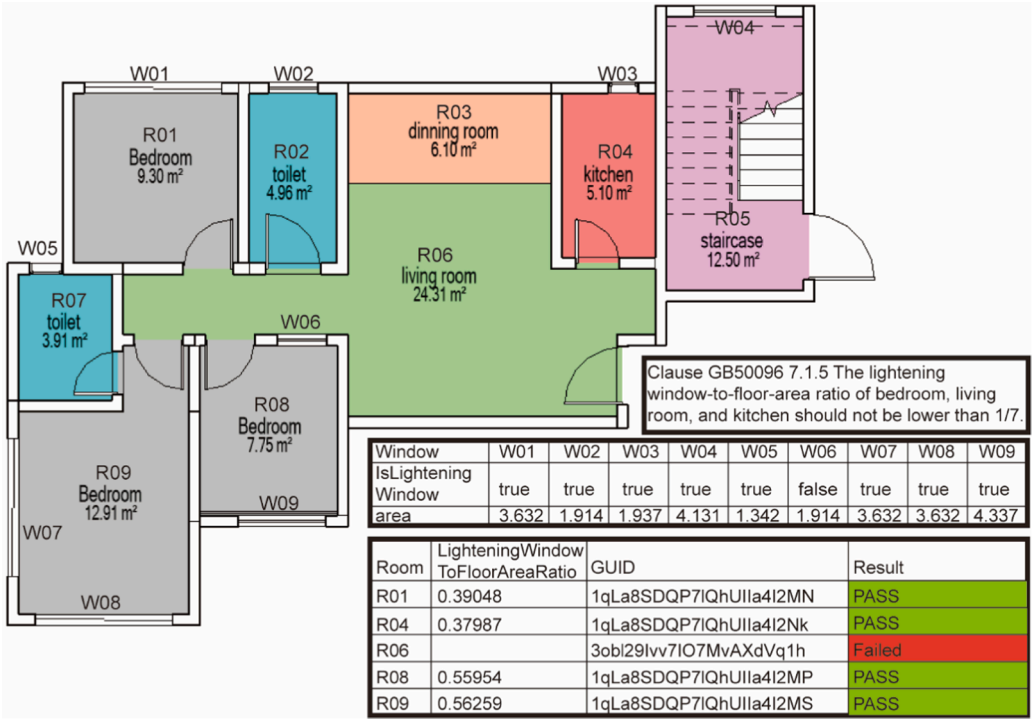
No.	Description	Mapping rules
6-1	Identify external window	① (?x rdf:type De:IfcWindow) -> (?x rdf:type Code:Window) ② (?x rdf:type De:IfcWallStandardCase) -> (?x rdf:type Code:Wall) ③ (?x rdf:type Code:Wall) (?x De:IsExternal "true"^^xsd:boolean) -> (?x rdf:type Code:ExternalWall) ④ (?x rdf:type Code:Window) (?y rdf:type Code:ExternalWall) (?y Code:relatedTo ?x) -> (?x rdf:type Code:ExternalWindow)
6-2	Identify stair railing	① (?x rdf:type De:IfcStair) -> (?x rdf:type Code:Stair) ② (?x rdf:type De:IfcRailing) -> (?x rdf:type Code:Railing) ③ (?x rdf:type Code:Railing) (?st rdf:type Code:Stair) (?st De:has ?x) -> (?x rdf:type Code:stairRailing)
6-3	Identify stair landing	① (?x rdf:type De:IfcStair) -> (?x rdf:type Code:Stair) ② (?x rdf:type De:IfcSlab) -> (?x rdf:type Code:Slab) ③ (?x rdf:type Code:Slab) (?st rdf:type Code:Stair) (?st De:has ?x) -> (?x rdf:type Code:StairLanding)

4.4. Discussion

The research efforts in this article are centered on facilitating the systematic implementation of automated code compliance checking based on BIM and ontology. The suggested structure of the automated code compliance checking platform includes five major function modules: (1) code ontology development module, which aims at defining knowledge scope and terminology for the code compliance checking; (2) designed model ontology converter module, which is utilized to simplify the data description structure for the designed model; (3) rules management module, which serves as the repository of the checking rules and mapping rules that are describing logics of the code compliance checking process; (4) ontology mapping and completeness checking module, which is used for mediation between designed model ontology and code ontology and for ensuring prerequisites of the following checking tasks; and (5) check reports generation module, which executes a rule-based reasoning process for code compliance checking.

Table 7
Comparison of different code compliance checking process framework.

Procedure		Executing time (s)	
		The proposed code compliance checking process	A normal code compliance checking process
Pre-processing	Converting IFC to ifcOWL	41.53	41.53
	Converting ifcOWL to the designed model ontology	62	\
Business procedure	Ontology reading	1	29
	Ontology mapping	0.786	20
	Pre-checking, rule-based checking and reports generation	15	220
Total		119.416	310.53



Window	W01	W02	W03	W04	W05	W06	W07	W08	W09
IsLightening Window	true	true	true	true	true	false	true	true	true
area	3.632	1.914	1.937	4.131	1.342	1.914	3.632	3.632	4.337

Room	LighteningWindow ToFloorAreaRatio	GUID	Result
R01	0.39048	1qLa8SDQP7lQhUIla4I2MN	PASS
R04	0.37987	1qLa8SDQP7lQhUIla4I2Nk	PASS
R06		3obl29lvv7lO7MvAXdVq1h	Failed
R08	0.55954	1qLa8SDQP7lQhUIla4I2MP	PASS
R09	0.56259	1qLa8SDQP7lQhUIla4I2MS	PASS

Fig. 15. Part of the checking results presented by the automated code compliance checking platform.

Compared to other research, the first contribution of the proposed platform is that a sub model of ifcOWL, namely the designed model ontology, could be created automatically based on modifying the local structure of ifcOWL during the checking process. The relations between IFC entities in the designed model ontology are simpler than those in the ifcOWL. As a result, file processing time may be saved while executing the code compliance checking process. In addition, a set of mapping rules are used to deal with the gaps between IFC entities and the design code concepts and provide semantic enrichment of the building model information. The information completeness of the building model could also be pre-checked before executing the code compliance checking procedure. Each checking clause is converted to the checking rule and the knowledge graph so that users could realize the checking logics and monitor the checking process directly. Besides, by decoupling the management of project data in the designed model converter process, ontology checking process, completeness checking process, and rule-based reasoning process, the proposed platform could also support multi-task checking (e.g., multi-code checking and multi-model checking).

5. Conclusions

Code compliance checking is a critical procedure in the AEC (Architecture, Engineering, and Construction) projects. This paper proposes an automated code compliance checking methodology based on BIM and ontology to reduce manual errors and make the checking process easier to understand and monitor. The proposed approach consists of a set of ontologies (i.e., code, designed model merged, and code compliance checking ontology) considered the knowledge bases. The knowledge bases are a list of rules (i.e., mapping rules and checking rules) that are deemed for providing expert logics, and a rule-based reasoning procedure that is utilized for checking reports generation. Based on the proposed approach, an automated code compliance checking platform including five functional subsystems, is established for case study implementation.

There are three main findings of this research. Firstly, pre-processing an ifcOWL ontology, such as element structure reorganization and element properties extraction, could reduce executing time and save running memories. Secondly, ontology mapping is a feasible way to address semantic gaps between IFC and regulatory documents and provide semantic enrichment of design information. Because of describing data as a relational graph, the checking rules could be converted into knowledge graphs so that domain experts or users could understand and monitor the checking process. Thirdly, the proposed platform could support the simple checking tasks and complicated checking tasks (i.e. multi-model checking and multi-code checking) by decoupling execution within the functional modules.

Further work could focus on two aspects; on the one hand, the

current method for rules development still relies on human effort. In the future, to enhance the developing efficiency and saving time–cost, natural language processing technologies and deep learning technologies could be applied to transform clauses into the knowledge graph and into the mapping rules and the checking rules automatically. On the other hand, this manuscript considers limited ranges of types of codes. For example, an extension of the proposed methodology is required for the codes that focus on procedural checking such as the construction schedule management. Besides, the mapping rules and the checking rule relevance to geometry remained to be figured out by external programming.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The author would like to extend their sincere gratitude to SADI Architectural Design Institute for providing the tested building models. The authors also want to express their sincere thanks to Natural Science Foundation of Chongqing, China.

References

[1] C. Eastman, J. Lee, Y. Jeong, J. Lee, Automatic rule-based checking of building designs, Automat. Constr. 18 (2009) 1011–1033, https://doi.org/10.1016/j.autcon.2009.07.002.
[2] T.F. Sing, Q. Zhong, Construction and real estate NETwork (CORENET), Facilities 19 (2001) 419–428, https://doi.org/10.1108/EUM0000000005831.
[3] Jotne EPM Technology. 2016. About Jotne IT. Retrieved from http://www.epmtech.jotne.com/about-jotne-it. Accessed 1 Feb 2020.
[4] J. Dimyadi, G. Governatori, R. Amor, 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), (2017), Heraklion, Greece: Heriot-Watt University, Edinburgh, UK, 10.24928/JC3-2017/0012.
[5] X. Tan, A. Hammad, P. Fazio, Automated code compliance checking for building envelope design, J. Comput. Civil Eng. 24 (2) (2010) 203–211, https://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203).
[6] J. Choi, J. Choi, I. Kim, Development of BIM-based evacuation regulation checking system for high-rise and complex buildings, Automat. Constr. 46 (2014) 38–49, https://doi.org/10.1016/j.autcon.2013.12.005.
[7] J. Melzner, S. Zhang, J. Teizer, H. Bargstädt, A case study on automated safety compliance checking to assist fall protection design and planning in building information models, Constr. Manage. Econ. 31 (2013) 661–674, https://doi.org/10.1080/01446193.2013.780662.
[8] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, Automat. Constr. 53 (2015) 69–82, https://doi.org/10.1016/j.autcon.2015.03.003.
[9] T. Bloch, R. Sacks, Clustering information types for semantic enrichment of building information models to support automated code compliance checking, J. Comput. Civil Eng. 34 (2020) 4020040, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000922.
[10] P. Pauwels, T.M. de Farias, C. Zhang, A. Roxin, J. Beetz, J. De Roo, C. Nicolle, A performance benchmark over semantic rule checking approaches in construction industry, Adv. Eng. Inform. 33 (2017) 68–88, https://doi.org/10.1016/j.aei.2017.05.001.
[11] Apache Jena, Retrieved from https://jena.apache.org/. Accessed 23 April 2021.
[12] A. Yurchyshyna, C. F. Zucker, N. Le Thanh, C. Lima, A. Zarli, Towards an ontology-based approach for conformance checking modelling in construction. In Proceedings of the 24th W78 Conference–Bringing ITC Knowledge To Work, (2007), Maribor, Slovenia.
[13] B. T. Zhong, L. Y. Ding, H. B. Luo, Y. Zhou, Y. Z. Hu, H. M. Hu, Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking, Automat. Constr. 28 (2012), 58-70, 10.1016/j.autcon.2012.06.006.
[14] K.R. Bouzidi, B. Fies, C. Faron-Zucker, A. Zarli, N.L. Thanh, Semantic web approach to ease regulation compliance checking in construction industry, Future Internet 4 (2012) 830–851, https://doi.org/10.3390/fi4030830.
[15] B. Zhong, C. Gan, H. Luo, X. Xing, Ontology-based framework for building environmental monitoring and compliance checking under BIM environment, Build. Environ. 141 (2018) 127–142, https://doi.org/10.1016/j.buildenv.2018.05.046.
[16] X. Xu, H. Cai, Semantic approach to compliance checking of underground utilities, Automat. Constr. 109 (2020), 103006, https://doi.org/10.1016/j.autcon.2019.103006.
[17] Y. Lu, Q. Li, Z. Zhou, Y. Deng, Ontology-based knowledge modeling for automated construction safety checking, Safety Sci. 79 (2015) 11–18, https://doi.org/10.1016/j.ssci.2015.05.008.
[18] T.H. Beach, Y. Rezgui, H. Li, T. Kasim, A rule-based semantic approach for automated regulatory compliance in the construction sector, Expert Syst. Appl. 42 (2015) 5219–5231, https://doi.org/10.1016/j.eswa.2015.02.029.
[19] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, Int. J. Hum.-Comput. St. 43 (1995) 907–928, https://doi.org/10.1006/ijhc.1995.1081.
[20] C. Pardo, F.J. Pino, F. García, M. Piattini, M.T. Baldassarre, An ontology for the harmonization of multiple standards and models, Comput. Stand. Inter. 34 (2012) 48–59, https://doi.org/10.1016/j.csi.2011.05.005.
[21] X. Xing, B. Zhong, H. Luo, H. Li, H. Wu, Ontology for safety risk identification in metro construction, Comput. Ind. 109 (2019) 14–30, https://doi.org/10.1016/j.compind.2019.04.001.
[22] Z. Ming, G. Sharma, J.K. Allen, F. Mistree, An ontology for representing knowledge of decision interactions in Decision-Based design, Comput. Ind. 114 (2020), 103145, https://doi.org/10.1016/j.compind.2019.103145.
[23] C. Mignard, C. Nicolle, Merging BIM and GIS using ontologies application to urban facility management in ACTIVe3D, Comput. Ind. 65 (2014) 1276–1290, https://doi.org/10.1016/j.compind.2014.07.008.
[24] R. Jardim-Goncalves, C. Coutinho, A. Cretan, C.F. Da Silva, P. Ghodous, Collaborative negotiation for ontology-driven enterprise businesses, Comput. Ind. 65 (2014) 1232–1241, https://doi.org/10.1016/j.compind.2014.01.001.
[25] Z. Huang, C. Jowers, A. Dehghan-Manshadi, M.S. Dargusch, Smart manufacturing and DVSM based on an Ontological approach, Comput. Ind. 117 (2020), 103189, https://doi.org/10.1016/j.compind.2020.103189.
[26] H. Schevers, R. Drogemuller, Converting the industry foundation classes to the web ontology language, in: Proceedings of the First International Conference on Semantics, Knowledge and Grid, 2005, https://doi.org/10.1109/SKG.2005.59.
[27] W. Terkaj, A. Šojić, Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology, Automat. Constr. 57 (2015) 188–201, https://doi.org/10.1016/j.autcon.2015.04.010.
[28] P. Pauwels, W. Terkaj, EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology, Automat. Constr. 63 (2016) 100–133, https://doi.org/10.1016/j.autcon.2015.12.003.
[29] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, J. Van Campenhout, A semantic rule checking environment for building performance checking, Automat. Constr. 20 (2011) 506–518, https://doi.org/10.1016/j.autcon.2010.11.017.
[30] J. Davies, R. Studer, P. Warren, Semantic Web technologies: trends and research in ontology-based systems, John Wiley & Sons, Ltd, 2006, ISBN: 9780470025963.
[31] M. H. Rasmussen, P. Pauwels, M. Lefrancois, G. F. Schneider, C. A. Hviid, J. Karlshøj, Recent changes in the Building Topology Ontology. In: Proceedings of the Linked Data in Architecture and Construction (LDAC 2017), (2017), Dijon, France, 10.13140/RG.2.2.32365.28647.
[32] W. Terkaj, P. Pauwels, A Method to generate a Modular ifcOWL Ontology. In: Proceedings of the the 8th Workshop Formal Ontologies Meet Industry, Joint Ontology Workshops 2017, (2017), Bolzano, Italy.
[33] E. V. Kalemi, F. Cheung, A. Tawil, P. Patlakas, IfcOWL-DfMA a new ontology for the offsite construction domain. In: Proceedings of the the 8th Linked Data in Architecture and Construction Workshop, (2020), Dublin, Ireland.
[34] J. Lee, C.M. Eastman, Y.C. Lee, Implementation of a BIM domain-specific language for the building environment rule and analysis, J. Intell. Robot. Syst. 79 (2015) 507–522, https://doi.org/10.1007/s10846-014-0117-7.
[35] H. Lee, J. Lee, S. Park, I. Kim, Translating building legislation into a computer-executable format for evaluating building permit requirements, Automat. Constr. 71 (2016), 49-61, 10.1016/j.autcon.2016.04.008.
[36] T.H. Beach, T. Kasim, H. Li, N. Nisbet, Y. Rezgui, Towards automated compliance checking in the construction industry. In International Conference on Database and Expert Systems Applications (pp. 366-380). Springer, Berlin, Heidelberg, 2013.
[37] S. Macit İlal, H.M. Günaydın, Computer representation of building codes for automated compliance checking, Automat. Constr. 82 (2017) 43–58, https://doi.org/10.1016/j.autcon.2017.06.018.
[38] N.F. Noy, D.L. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. 2001.
[39] P. Pauwels. 2020. IFCtoRDF. Retrieved from https://github.com/pipauwel/IFCtoRDF. Accessed 15 March 2021.
[40] T.M. de Farias, A. Roxin, C. Nicolle, SWRL rule-selection methodology for ontology interoperability, Data Knowl. Eng. 105 (2016) 53–72, https://doi.org/10.1016/j.datak.2015.09.001.
[41] ifcOWL. Retrieved from https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/. Accessed 03 February 2021.
[42] N.O. Nawari, A generalized adaptive framework (GAF) for automating code compliance checking, Buildings (Basel) 9 (2019) 86, https://doi.org/10.3390/buildings9040086.

[43] Z. Pan, J. Shi, L. Jiang, A novel HDF-Based data compression and integration approach to support BIM-GIS practical applications, Adv. Civil Eng. 2020 (2020) 1–22, https://doi.org/10.1155/2020/8865107.
[44] W. Solihin, C. Eastman, A Knowledge Representation Approach to Capturing BIM Based Rule Checking Requirements Using Conceptual Graph. In Proceedings of the 32nd International Conference on Application of IT in the AEC industry 2015 (CIB W78 Conference 2015), (2015), Eindhoven, The Netherlands, URL: https://itc.scix.net/pdfs/w78-2015-paper-071.pdf.
[45] J. Dimyadi, G. Governatori, R. Amor. Evaluating LegalDocML and LegalRuleML as a Standard for Sharing Normative Information in the AEC/FM Domain. In Processdings of the Joint Conference on Computing in Construction (JC3), (2017), Heraklion, Crete, Greece, 10.24928/JC3-2017/0012.