applied sciences

Article

Combined MvdXML and Semantic Technologies for Green Construction Code Checking

Shaohua Jiang [1,*], Zheng Wu [1], Bo Zhang [1] and Hee Sung Cha [2]

[1] Department of Construction Management, Faculty of Infrastructure Engineering, Dalian University of Technology, Dalian 116024, China; wuzheng0709@gmail.com (Z.W.); zhangbo5951@163.com (B.Z.)
[2] Department of Architectural Engineering, Ajou University, Suwon 16499, Korea; hscha@ajou.ac.kr
* Correspondence: shjiang@dlut.edu.cn; Tel.: +86-411-8470-7482

Received: 31 January 2019; Accepted: 27 March 2019; Published: 8 April 2019

Abstract: The construction process plays a key role in sustainable development of the environment. With the concept of sustainable construction being put forward in the world, some countries released green construction standards to strengthen the requirements in the construction phase. Green construction code checking needs to integrate semantic information embedded in green construction standards and model information involved in Industry Foundation Classes (IFC) and/or Model View Definition (MVD), which are generated separately and lead to difficulty in information integration for green construction code checking. At present, the existing code-checking methods cannot be directly used for green construction. Related practitioners need an efficient and convenient method for green construction code checking urgently. To ameliorate this situation, this research proposes an innovative approach to organize, store, and re-use green construction knowledge by combining mvdXML and semantic technologies. The code checking of green construction is classified into four types based on the difficulty level to meet the requirements of the clauses in green construction standard. Depending on the characteristics of each inspection type, mvdXML or semantic technology is adopted for the appropriate inspection type. This paper demonstrates the deployment and validation of such automated checking procedures in a case study. Based on these experiences, a detailed discussion about the identified issues is provided as the starting point for future research.

Keywords: green construction; code checking; mvdXML; semantic technology

1. Introduction

As people pay more and more attention to the living environment, the concept of green construction is becoming widespread [1]. According to statistics, a great level of pollution and waste occur in the different phases of a building's life cycle [2]. Most of these activities take place in the construction phase; thus, green construction is a very important strategy of sustainable development [3]. For example, waste generated in construction and demolition processes comprised around 50% of the solid waste in South Korea in 2013 [4]. Many cases show that the code checking based on building information modeling (BIM) is an effective means to reduce the amount of construction waste, since it is mainly generated due to improper design and unexpected changes in the design and construction phases. In response to this reality, some countries such as America and China released green construction-related standards and codes. The definitions of green construction in different countries are similar. In China, green construction is defined as "under the premise of guaranteeing the basic requirements of quality and safety, the construction activities which should be adopted to maximize resource conservation and minimize the negative impact on the environment through scientific management and technological progress, and to achieve the goal of four savings (energy, land, water, and materials) and environmental protection", which is specified in the Code for Green Construction of Building (GB/T50905-2014) issued by the Ministry of Housing and Urban-Rural Development of the People's Republic of China (MOHURD) in 2014.

The evaluation of green construction is becoming more and more complex; however, traditional evaluation processes need a lot of manpower, materials, and time, which will undoubtedly increase the unreliability and bring uncertainty to the quality of construction project. Therefore, construction industry practitioners put forward urgent requirements for a more efficient approach of code checking of green construction.

Some code-checking cases are shown in other researchers' papers. For example, Won et al. showed that BIM-based validation could prevent 4.3–15.2% of construction waste that might have been generated without using BIM [4]. However, the requirements of green construction were not fully considered. Therefore, existing approached cannot be directly used for green construction code checking, which results in an imperative need for innovative technology to improve the current condition.

Green construction code checking requires distinct domain knowledge and building model information. Model view definition (MVD), proposed by buildingSMART, defines a subset of the IFC schema needed to satisfy data exchange requirements, and it helps software vendors develop IFC import and export features that allow project participants to share and exchange BIM information. It is a kind of handshake protocol that allows expected meaningful requirements, implementation, and results. The method to publish the concepts and associated rules is called mvdXML [5]. Ontology and semantic web technology offer an opportunity to enable domain knowledge to be represented semantically. Semantic Query Web Rule Language (SQWRL) is a Semantic Web Rule Language (SWRL)-based query language that provides SQL(Structured Query Language)-like operators for extracting information from ontologies in Web Ontology Language (OWL), which is also widely used for code checking [6,7]. The goal of the mvdXML and semantic technologies is not only to put forward an automated knowledge framework, but also to offer the impetus to gear up the Architecture, Engineering and Construction (AEC) industry toward greater interoperability through the deployment of the BIM-based code checking.

However, both mvdXML and semantic technologies have limitations. This paper combines mvdXML and semantic technologies to transfer green construction standards to computer-interpretable code. This approach allows researchers to target critical needs across application areas and to solve the problems of code checking.

2. Research Background

Three levels of BIM implementation are proposed in the Bew–Richards maturity model, in which a code-checking process is implemented in Level 2 and can be continually developed with more functions in Level 3 [8]. Many studies were conducted on code checking, which can be categorized into four classes based on the adopted approaches. In this section, the importance of green construction, particularly green construction code checking, is summarized, and previous code-checking approaches are reviewed [9–11].

2.1. Importance of Green Construction

With the acceleration of the urbanization process and the rapid development of the social economy, the business scope of the construction industry is constantly expanding. At the same time, the adverse impact of waste and pollution caused by construction activities on the environment cannot be ignored, especially in the construction stage. In recent years, people's awareness of environmental protection and resource conservation increased; thus, the concept of sustainable development was widely acknowledged. Green construction is the embodiment of the concept of sustainable development in the construction industry. Only by paying attention to the dissemination of the green construction concept and the application of green construction technology can the sustainable development of the construction industry be realized.

2.2. Green Construction Code Checking

Knowledge and rules are two critical parts in the green construction code-checking process. The knowledge related to the green construction code and the project-specific information required by green construction code checking are scattered and fragmented. The rules in the green construction code-checking process should be interpreted systematically and made tractable, whereby complete green construction rule interpretation involves knowledge from human experts in many professional fields. Therefore, the exact requirements for green construction code checking are often much more complex than other standards. Because much information relevant to green construction standard is not available in the BIM model, the IFC standard needs to be physically extended to express the missing information. As not everything in the BIM can be defined at the initial stage, some missing, evolving, or additional information can be inserted into the mvdXML, allowing the mvdXML to include the original IFC information and the extended information.

Some scholars systematically analyzed code-checking-related systems, concept libraries, query languages, reasoners, and model view definitions [12,13]. From their analysis, we can see that there are many kinds of rules and concepts supported by the information included in different levels of detail (LOD). In traditional rule checking or code checking, LOD 300 is assumed in general, but green construction code checking needs higher LOD. Before green construction code checking, the BIM model should be modeled as specific assemblies, with complete fabrication, assembly, and detailing information, in addition to precise quantity, size, shape, location, and orientation. Non-geometric information of the model elements can also be attached, which includes model details and elements that represent how building elements interface with various systems and other building elements with model view. Requirements for code checking should match with all kinds of green construction-related information in BIM. In this case, LOD 400 is expected for green construction code checking.

2.3. Code-Checking Approaches in Construction Industry

There are four types of code-checking approaches, i.e., code checking based on existing BIM software, code checking based on object-oriented approach, code checking based on logical rules, and code checking approach based on semantic web. Their characteristics and limitations are summarized in this section.

The code-checking approach based on existing BIM software and code checking based on an object-oriented approach are usually used for simple quantitative model checking, and they both have less scalability. The code-checking approach based on logical rules and the code-checking approach based on semantic web are both widely used. MvdXML technology is a code-checking approach based on logical rules. Fahad et al. compared mvdXML technology and semantic web technology to deal with abstract concepts or physical objects, and pointed out that each method has its pros and cons and can be used according to the requirements of the research project [14]. The mvdXML method is an appropriate choice for simple logical rules. In the implementation of mvdXML method, the BIM Collaboration Format (BCF) is used to report identified issues. BCF is an open standard originally proposed by buildingSMART to enable workflow communication between different applications. Unlike traditional text-based reports, mvdXML method links communication to model view, and it can also be easily implemented by visualization applications. The code-checking approach based on semantic web can perform complicated semantic inferences which cannot be conducted in the mvdXML approach; therefore, it is also widely used.

Based on the particularity and requirements of green construction code checking, we combined mvdXML technology and semantic web technology to conduct green construction code checking. Therefore, we mainly introduce the latter two approaches here.

2.3.1. Code Checking Based on Logical Rules

The most common language in interpreting the natural language is first-order predicate logic. Choi et al. created links between the Standard for the Exchange of Product Model Data (STEP) and Extensible Markup Language (XML) formats to develop a standardized automated inspection system to share architectural drawings and document information [15]. MvdXML is an XML format file including MVD information. BIM Rule Language (BIMRL) was developed by SQL to support code checking, which is a query and manipulation language used for managing the rule data [16]. Nawari et al. demonstrated the construction plan with the BIM model and used the model-checking software to achieve automatic inspection of the normative terms, while the specification and standards were in monotonous and rigid formats [17]. Nawari et al. pointed out that the implementation of the automatic specification check is based on two main parts; one is to convert building code to computable rules, i.e., Smart Codes, the other is to use the BIM model to ensure the level of detail [18]. Ilhan et al. proposed a framework for providing an integrated platform to facilitate the green code generation. The proposed model helps design team-assigned sustainable properties and helps assess sustainability performance of a project during the design stage so that decisions can be made on time for Building Research Establishment Environmental Assessment Methodology (BREEAM) certification [19]. Ciribini et al. implemented an interoperable IFC-based process to support code checking through four-dimensional (4D) BIM; an auto-matching between BIM objects and construction activities was achieved [20]. Zhang et al. developed algorithms that automatically analyze a building model to detect safety hazards and suggest preventive measures to users for different cases involving fall-related hazards. A rule-based engine was implemented on the top of a commercially available BIM platform to show the feasibility [21]. Kasim et al. presented a generic approach for automating BIM compliance, checking with standards and best practice with a focus on sustainability-related standards. A methodology was presented to provide a reusable solution for compliance checking within a similar context. The proposed approach was generic in nature, thus allowing implementation in different fields of engineering including electrical engineering, energy, water, and electronic engineering. In this approach, the RASE (Requirement, Applicability, Selection, Exception) methodology was utilized to extract requirements from sustainability-based regulations, with the goal of converting them into coded rules for execution by a rule engine [22]. Lee et al. suggested a robust MVD validation process using a modularized validation platform that evaluates an IFC instance file according to diverse types of rule sets of MVDs. This research employed the MVD of the precast concrete domain using the identified rule logic and checking structures to extract rule templates that consisted of 10 types of rule logic [23]. Dimyadi et al. proposed a formalized way to capture these requirements using a graphical notation to represent code rules in a machine- and human-readable language, which allowed unambiguous description of the requirements. Such unambiguous description could be understood by all the participants in the implementation efforts [24,25]. Zhang et al. proposed an automated compliance checking system for automatically extracting information from construction regulatory textual documents and transforming them into logic clauses using semantic Natural Language Processing (NLP) and logic inference, which focused on presenting quantitative requirements and the corresponding algorithms. The system could be extended to support the checking of different types of existential requirements containing certain building elements in rules [26–28].

2.3.2. Code Checking Based on Semantic Web

There are some building codes computerizing systems based on ontology-based approaches and semantic web information. Ontology-based approaches focus on formalizing conformance requirements, which are conducted under knowledge extraction and semantic mapping [29]. Semantic web information is about how to enhance the IFC model using description language, which is based on a logic theory. Pauwels et al. took advantage of the logical basis of Resource Description Framework (RDF), using semantic web technologies to support data interoperability and integrate relationships between semantic information and model information [30,31]. The semantic web was built in a semantic network on the World Wide Web (WWW). In semantic web, objects and logical relationships between objects are displayed by graph. The Resource Description Framework (RDF) is a standard model for data interchange on the web [32,33]. RDF is used for representing the graph structure, which provides expressions and statements to describe relationships between resources. RDF extends the linking structure of the web to use Unique Resource Identifiers (URIs) to name the relationship between things, as well as the two ends of the link. Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications. RDF is used in Web Ontology Language (OWL) to describe more complex ontologies [34,35].

The ontological approach was also adopted by International Code Council (ICC) through the development of SMARTcodes in 2006. ICC establishes an International Energy Conservation Code (IECC) dictionary, which is a method of communication between regulations and building models [36]. In addition to IECC, Bakillah et al. proposed a new system that supports geospatial data retrieval from multiple and complementary sources [37]. The system uses the SQWRL to support inference with complex queries and to enable combination of complementary and heterogeneous data. Zhong et al. explored an ontology-based semantic modeling approach of regulation constraints and proposed a meta model for construction quality inspection and evaluation, i.e., CQIEOntology. Based on CQIEontology, the regulation constraints were modeled into OWL axioms and SWRL rules. Using these OWL axioms and SWRL rules, the regulation provisions imposed on construction quality inspection can be translated into a set of inspection tasks, before being associated with the specific construction tasks [38]. Lu et al. extracted information from construction safety regulations and represented constrain rules with SWRL. The construction safety-checking processes were implemented in the JESS (Java Expert Shell System) engine [39]. Ding et al. took advantage of the strength of BIM, ontology, and semantic web technology to establish an ontology-based framework for construction risk knowledge management in a BIM environment [40]. Lee et al. proposed an ontology-based framework which would help accurately recognize domain knowledge and appropriate requirements for developing reusable concept modules [41]. Zhang et al. proposed a pragmatic method to check real-world-scale BIM models, which transforms BIM models into a well-defined OWL model. Rules were formalized by a structured natural language (SNL) designed intentionally to describe building regulations. The checking engine was based on Simple Protocol and RDF Query Language (SPARQL) queries on OWL models, which can effectively improve the time efficiency and deal with large-scale applications [42]. De Farias et al. combined the ifcOWL ontology with SWRL rules and performed a more intuitive and flexible extraction of building views than the MVD approach [43].

The objective of this study was to propose an approach to organize, store, and re-use green construction knowledge. Therefore, considering the particularity of green construction standards and the limitation in current code-checking approaches, in this paper, BIM, logical rule, ontology, and semantic web query language were integrated to conduct green construction code checking. The conceptual model ontology, work item ontology, and construction condition ontology for green construction were built, while the missing information for green construction code checking was added using IfcDoc, and various kinds of green construction rules were checked using an approach combining mvdXML and semantic technologies. Users can modify the parameters of each rule to match their local regulations using the approach proposed in this paper. Once the rule structure is defined, it is available for multiple code-checking projects.

3. Classification of Code Checking for Green Construction

With the development of green building certification, some national organizations and institutions further infiltrated the issues related to the construction phase to make up for the insufficiency of the requirements regarding green construction in green building certification. To reflect the environmental problems and guide the environmental management on the construction site, at present, some countries such as America and China released green construction-related standards and codes.

The code checking of green construction standards can be categorized into the following four types based on the difficulty level to meet the requirements of the clauses.

Class 1: Code checking based on simple defined attribute values in IFC schema;
Class 2: Code checking based on the extension of IFC entities;
Class 3: Code checking based on simple conceptual reasoning;
Class 4: Code checking based on construction condition reasoning.

This paper takes green construction standards and codes in China as an example.

In China, sustainability was widely debated in the construction industry in recent years. In order to strengthen the "green" development during the construction phase, MOHURD organized research efforts to conduct green construction research, issuing the "Evaluation Standard for Green Construction of Buildings" in 2010 (GB/T506040), which was promulgated to promote green construction, and releasing the "Code for green construction of building" in 2014 (GB/T50905-2014), which provides directional guidance for green construction.

According to the above classification of code checking, some example clauses in GB/T50905-2014 are shown in Table 1.

Table 1. Classification of various kinds of clauses in GB/T50905-2014.

Classification of Clauses	Types of Inspection	Example Clauses in GB/T50905-2014
Class 1	Code checking based on simple defined attribute value in IFC schema.	&3.2.1 Building material storage site should be no more than 500 km from construction site.
Class 2	Code checking based on the extension of IFC entities.	&4.0.5 A database of building materials should be established and a building material with good green performance should be adopted.
Class 3	Code checking based on simple conceptual reasoning	&9.2.4 Cast-in-place concrete raw materials should be mixed in construction site.
Class 4	Code checking based on construction condition reasoning	&9.2.7 When spraying materials on site, the ambient temperature should be 10–40 °C

In the case study, for each type of rule, an example from these standards with logic formulas was given using mvdXML and semantic technologies to specify its semantics.

4. Methodology of Green Construction Code Checking

Each method has its own characteristics and should be used according to requirements of the green construction specification. XML and mvdXML data can be transformed into OWL/RDF description language as the rule base of ontology (see Figure 1), which allows mvdXML and semantic technologies to be integrated for code checking. Based on this concept, an approach combining mvdXML and semantic technologies for code checking of green construction is introduced below (see Figure 2).

It is necessary to follow some specific steps for high efficiency and performance of code checking.

(1) Data source. The green construction standard is selected as one data source, which can be converted into XSLT (Extensible Stylesheet Language Transformations)/XML based on rule expression and classification of clauses. Model information in IFC and/or MVD is selected as another data source, which can be expressed in Excel and transformed to mvdXML form using the mvdXML rule transformer by data preprocessing [44]. The missing information for green construction code checking can be added in mvdXML using IfcDoc.

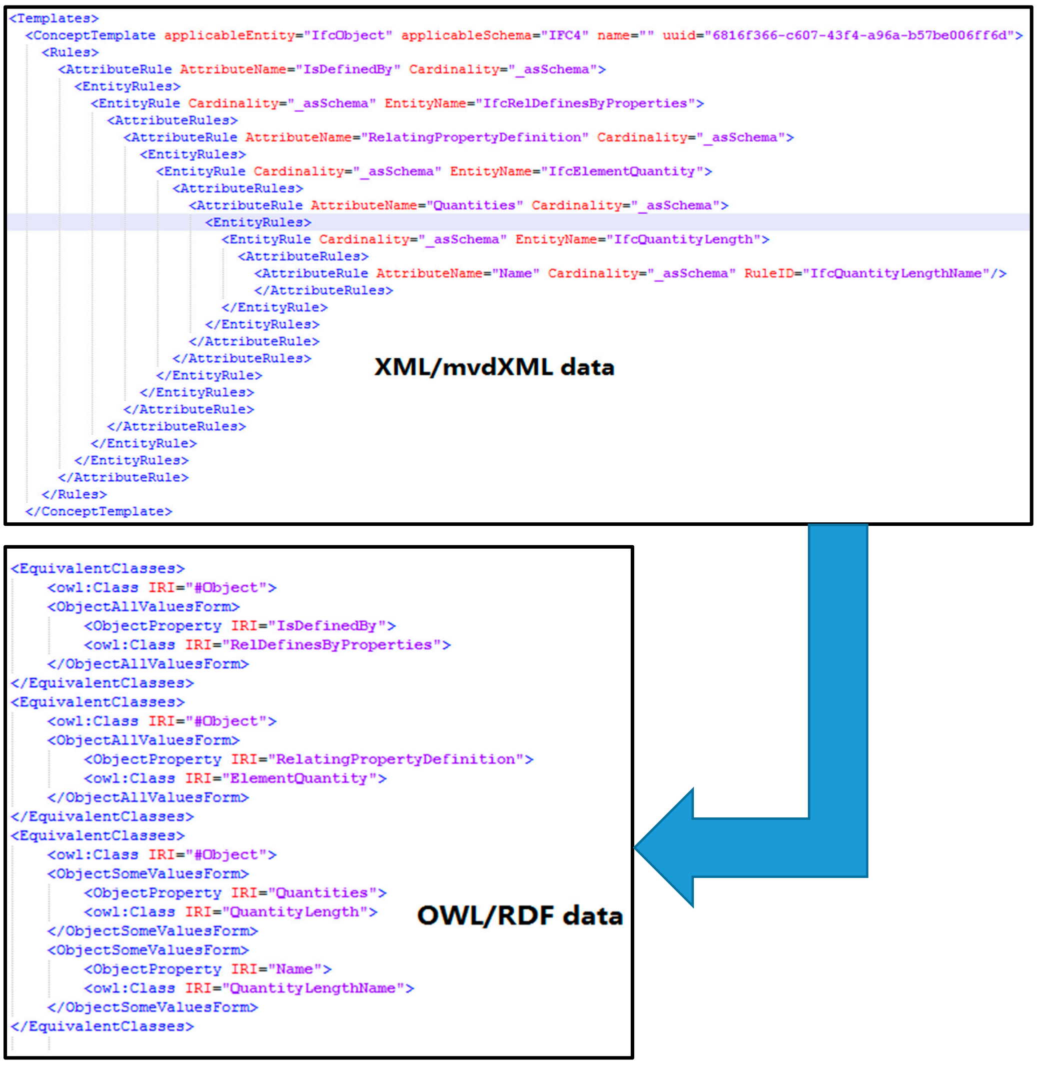
(2) Knowledge base. The conceptual model ontology, work item ontology, and construction condition ontology for green construction are built in this part. The ontologies in the knowledge base consist of facts and axioms.

(3) Rule base. The rule base includes customized rules that are used in conjunction with the content in the knowledge base for the JESS inference. The customized rules are based on the model information in IFC and/or MVD. The rules in this study were established in Protégé 3.4, and the grammar format was based on RDF [45].

(4) JESS inference. The facts and the rule base are matched to obtain inference results through JESS inference engine. After the transformation from the JESS rule language to RDF/OWL, the inferred facts can be used to update the knowledge base.

(5) Query. To meet the need of query function, the result needs to be transformed into a computer-readable format. The converted data representation is in the form of a triplet, which consists of resources, attributes, and values that can be represented by the concept and relationship of ontology and can be queried by SQWRL or SWRL. Finally, the report of code checking for green construction can be produced using Hypertext Markup Language (HTML).

In addition, the IFC model can be checked by the mvdXML ruleset and a BCF report is generated using a BCF generator; then, the issues can be shown in Revit using BCFier with a BIM view [46].


```
<Templates>
  <ConceptTemplate applicableEntity="IfcObject" applicableSchema="IFC4" name="" uuid="6816f366-c607-43f4-a96a-b57be006ff6d">
    <Rules>
      <AttributeRule AttributeName="IsDefinedBy" Cardinality="_asSchema">
        <EntityRules>
          <EntityRule Cardinality="_asSchema" EntityName="IfcRelDefinesByProperties">
            <AttributeRules>
              <AttributeRule AttributeName="RelatingPropertyDefinition" Cardinality="_asSchema">
                <EntityRules>
                  <EntityRule Cardinality="_asSchema" EntityName="IfcElementQuantity">
                    <AttributeRules>
                      <AttributeRule AttributeName="Quantities" Cardinality="_asSchema">
                        <EntityRules>
                          <EntityRule Cardinality="_asSchema" EntityName="IfcQuantityLength">
                            <AttributeRules>
                              <AttributeRule AttributeName="Name" Cardinality="_asSchema" RuleID="IfcQuantityLengthName"/>
                              </AttributeRules>
                          </EntityRule>
                        </EntityRules>
                      </AttributeRule>
                    </AttributeRules>
                  </EntityRule>
                </EntityRules>
              </AttributeRule>
            </AttributeRules>
          </EntityRule>
        </EntityRules>
      </AttributeRule>
    </Rules>
  </ConceptTemplate>
```

```
<EquivalentClasses>
    <owl:Class IRI="#Object">
    <ObjectAllValuesForm>
        <ObjectProperty IRI="IsDefinedBy">
        <owl:Class IRI="RelDefinesByProperties">
    </ObjectAllValuesForm>
</EquivalentClasses>
<EquivalentClasses>
    <owl:Class IRI="#Object">
    <ObjectAllValuesForm>
        <ObjectProperty IRI="RelatingPropertyDefinition">
        <owl:Class IRI="ElementQuantity">
    </ObjectAllValuesForm>
</EquivalentClasses>
<EquivalentClasses>
    <owl:Class IRI="#Object">
    <ObjectSomeValuesForm>
        <ObjectProperty IRI="Quantities">
        <owl:Class IRI="QuantityLength">
    </ObjectSomeValuesForm>
    <ObjectSomeValuesForm>
        <ObjectProperty IRI="Name">
        <owl:Class IRI="QuantityLengthName">
    </ObjectSomeValuesForm>
</EquivalentClasses>
```

Figure 1. Transform between XML/mvdXML and OWL/RDF data.

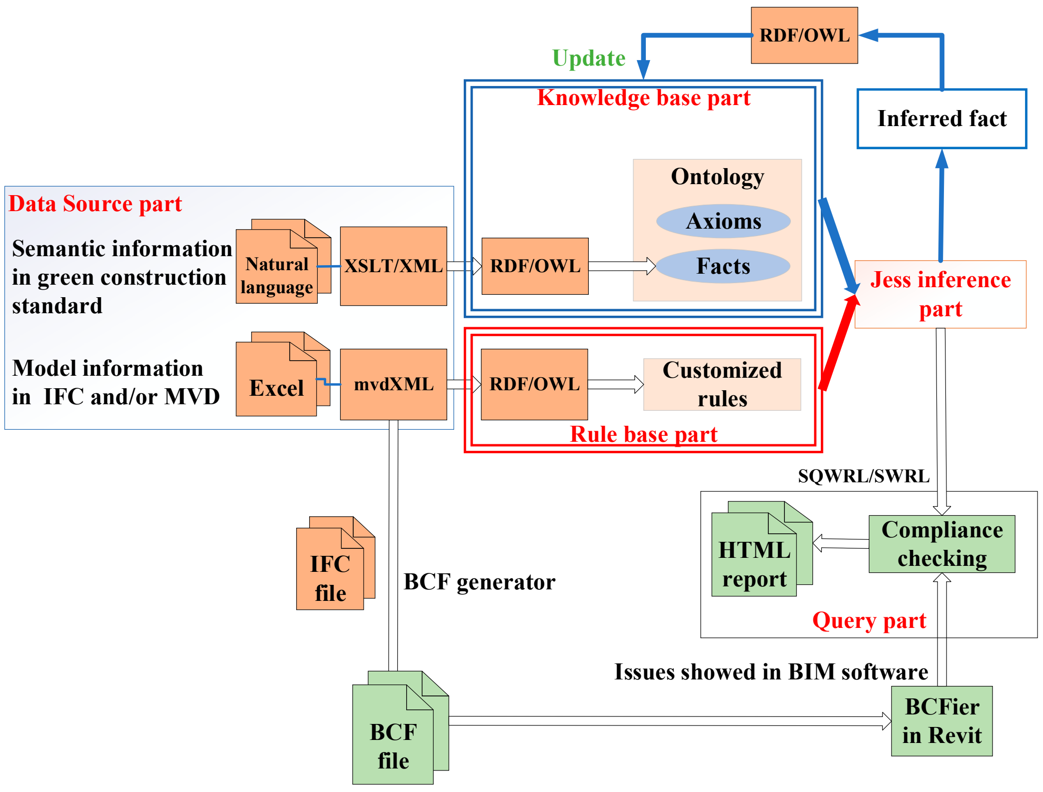


Figure 2. Workflow of code checking.

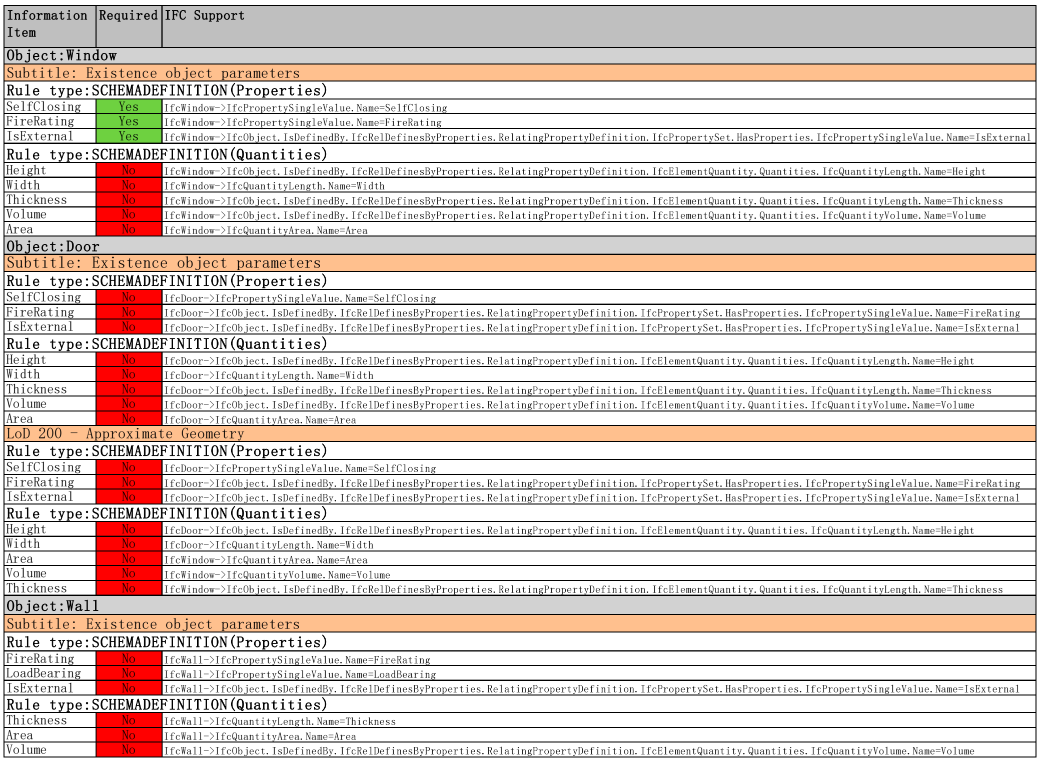
4.1. *MvdXML Technology for Code Checking*

Model view definitions (MVDs) are encoded in a format called mvdXML, which defines allowable values at particular attributes of particular data types [47]. For example, an MVD may require a wall providing a fire rating, a classification according to Table 22 of OmniClass, and information required for structural analysis, such as the elastic modulus of materials. In simple cases, mvdXML may define a single attribute on a single data type, while more complex cases may consist of graphs of objects and collections which can be defined by semantic technology. To check the rules in mvdXML, in general, four steps are needed: (1) generating IFC models using BIM-related software and model information in IFC and/or MVD expressed in Excel; (2) transforming Excel to mvdXML form using an mvdXML rule transformer by data preprocessing and inserting the information which is missing, evolving, or new in the mvdXML using IfcDoc. The IfcDoc tool is used to generate documentation of the IFC specification itself, the baseline MVD ConceptTemplates, and concepts that are part of the IFC specification. It, therefore, allows quickly expanding the generic MVD concepts to cover the specific requirements and business rules that are introduced by a set of exchange requirements; (3) evaluating and comparing IFC and mvdXML files [45]; (4) reading the output of the BCF generator using BIM software.

A checker was developed based on the open standard mvdXML as the format for structuring checking rules and the BIM Collaboration Format (BCF) to issue reports of the checking process. Two BIM operational standards, i.e., mvdXML and BCF were used for green construction code checking [46].

4.1.1. Data Source

The data source was an Excel spreadsheet. Figure 3 gives an overview of the data source. The data source contains rules that can validate if an object (e.g., window) contains certain properties and quantity parameters (e.g., self-closing). The property and quantity rules are often used in model checking.



Information Item	Required	IFC Support
Object:Window		
Subtitle: Existence object parameters		
Rule type:SCHEMADEFINITION(Properties)		
SelfClosing	Yes	IfcWindow->IfcPropertySingleValue.Name=SelfClosing
FireRating	Yes	IfcWindow->IfcPropertySingleValue.Name=FireRating
IsExternal	Yes	IfcWindow->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=IsExternal
Rule type:SCHEMADEFINITION(Quantities)		
Height	No	IfcWindow->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Height
Width	No	IfcWindow->IfcQuantityLength.Name=Width
Thickness	No	IfcWindow->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Thickness
Volume	No	IfcWindow->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityVolume.Name=Volume
Area	No	IfcWindow->IfcQuantityArea.Name=Area
Object:Door		
Subtitle: Existence object parameters		
Rule type:SCHEMADEFINITION(Properties)		
SelfClosing	No	IfcDoor->IfcPropertySingleValue.Name=SelfClosing
FireRating	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=FireRating
IsExternal	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=IsExternal
Rule type:SCHEMADEFINITION(Quantities)		
Height	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Height
Width	No	IfcDoor->IfcQuantityLength.Name=Width
Thickness	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Thickness
Volume	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityVolume.Name=Volume
Area	No	IfcDoor->IfcQuantityArea.Name=Area
LoD 200 – Approximate Geometry		
Rule type:SCHEMADEFINITION(Properties)		
SelfClosing	No	IfcDoor->IfcPropertySingleValue.Name=SelfClosing
FireRating	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=FireRating
IsExternal	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=IsExternal
Rule type:SCHEMADEFINITION(Quantities)		
Height	No	IfcDoor->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Height
Width	No	IfcDoor->IfcQuantityLength.Name=Width
Area	No	IfcWindow->IfcQuantityArea.Name=Area
Volume	No	IfcWindow->IfcQuantityVolume.Name=Volume
Thickness	No	IfcWindow->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityLength.Name=Thickness
Object:Wall		
Subtitle: Existence object parameters		
Rule type:SCHEMADEFINITION(Properties)		
FireRating	No	IfcWall->IfcPropertySingleValue.Name=FireRating
LoadBearing	No	IfcWall->IfcPropertySingleValue.Name=LoadBearing
IsExternal	No	IfcWall->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=IsExternal
Rule type:SCHEMADEFINITION(Quantities)		
Thickness	No	IfcWall->IfcQuantityLength.Name=Thickness
Area	No	IfcWall->IfcQuantityArea.Name=Area
Volume	No	IfcWall->IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcElementQuantity.Quantities.IfcQuantityVolume.Name=Volume

Figure 3. Excel data source.

The data source described in Figure 3 consists of three columns. The first column "Information Item" classifies the rule type and can be used to append a name to each rule. The second column "Required" specifies whether the rule should be converted by the mvdXML rule transformer. Each row should specify if the rule should be transformed into mvdXML format. Thus, "Yes" means that the mvdXML rule transformer should convert the rule to mvdXML, and "No" means the opposite. The third column "IFC Support" is a path that is interpreted by the mvdXML rule transformer. The section "IFC Support" elaborates how to create and adjust IFC support path.

4.1.2. MvdXML Rule Transformer

The mvdXML rule transformer is a tool for the generation of the mvdXML ruleset (see Figure 4). MvdXML rules are based on the open mvdXML standard. The open mvdXML standard ensures easy access and extensions of the ruleset by users. The mvdXML rule transformer can generate rules for all rule types defined in mvdXML schema. If there is some information that is missing, evolving, or new in the mvdXML, IfcDoc allows additional content to be created in IFC baseline, which can be transformed to mvdXML files. IfcDoc is divided into four submenus as follows:

(1) Documentation: content only affecting documentation output such as terms, references, and examples.
(2) Schema: content describing underlying data structures such as entities, enumerations, selections, and defined types added within a schema (data schemas, computer interpretable listings, alphabetical listings, and inheritance listings in IfcDoc).
(3) User data: content describing dynamic data structures such as property sets and quantity sets, added within a schema (data schemas, computer interpretable listings, alphabetical listings, and inheritance listings in IfcDoc).
(4) Model views: content describing model view definitions such as templates (concepts in IfcDoc), model views and exchanges (scope in IfcDoc), and concepts (underneath entities within data schemas, computer interpretable listings, alphabetical listings, and inheritance listings in IfcDoc).

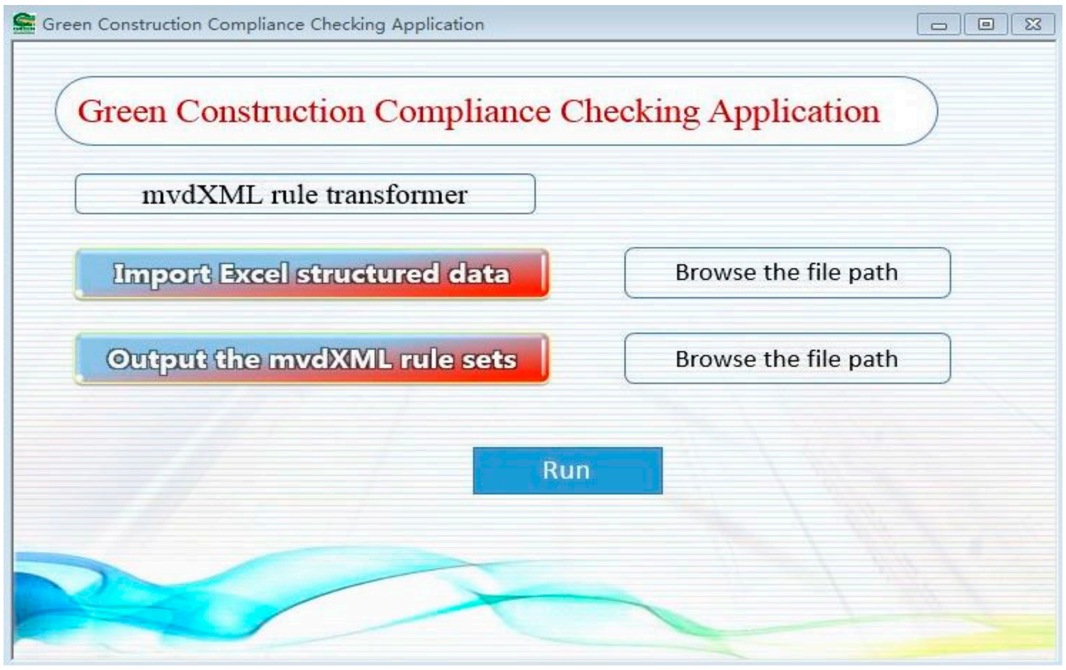

Green Construction Compliance Checking Application
Green Construction Compliance Checking Application
mvdXML rule transformer
Import Excel structured data
Browse the file path
Output the mvdXML rule sets
Browse the file path
Run

Figure 4. MvdXML rule transformer.

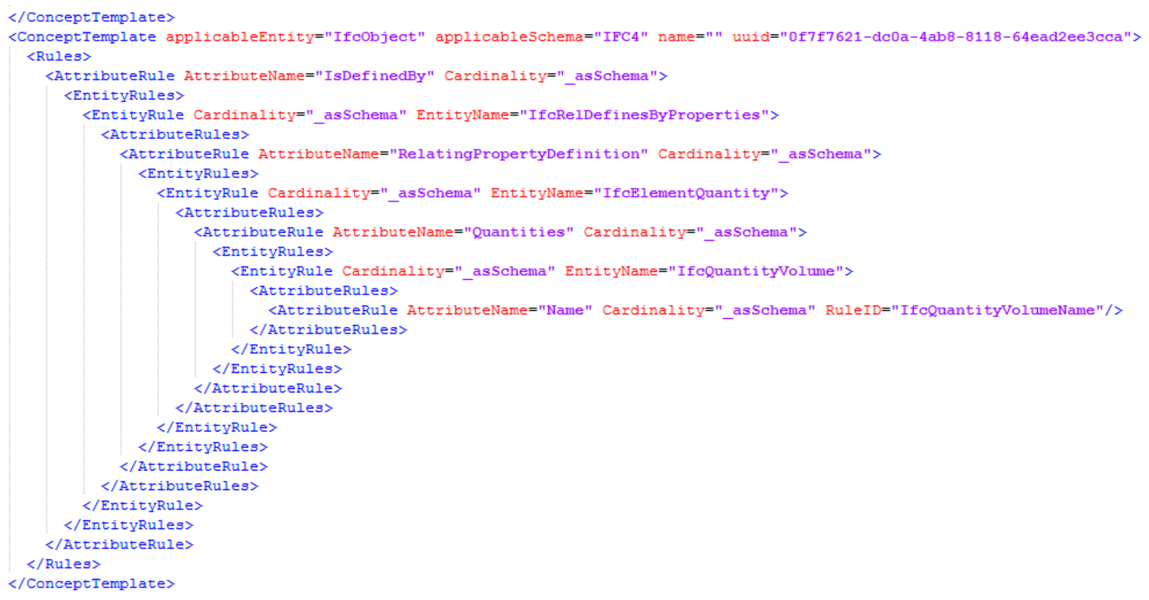
In this paper, the information related to green construction standards not defined in the existing IFC was extended in the case study.

4.1.3. MvdXML Rules

To use the mvdXML rule transformer, basic knowledge about the structure of mvdXML files is required. The <ConceptTemplate> element defines the structure that should be complied with by related concepts (see Figure 5). For instance, Concept Template is able to assign IfcPropertySingleValues to all subtypes of an IfcObject. Concept Template defines rules for the entire IfcObject based on IFC2X4. The elements between <Rules> define the path to the ApplicableEntity (i.e., IfcPropertySingleValueName). The universally unique identifier (UUID) of the Concept Template is an essential element which is generated to relate concepts to the Concept Template. For instance, Figure 6 is the mvdXML snippet that shows the definition of the "external flag" concept of "external loadbearing wall", and Figure 7 is the mvdXML snippet that shows the definition of "applicability". The name of ConceptRoot element can be used to search for the name of a concept [47,48]. Three parts are relevant in the ConceptRoot definition.

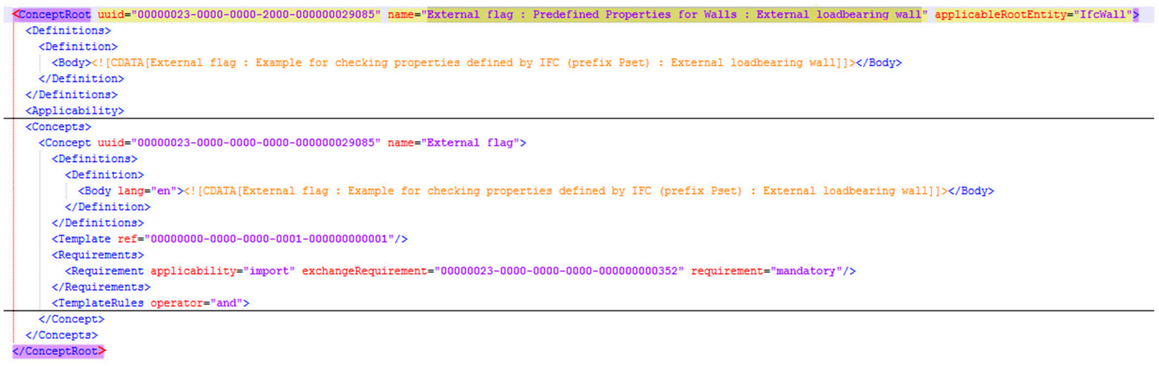
(1) ApplicableRootEntity and the applicability element (see Figure 6) define all IfcWall instances that fulfill the applicability constraints.
(2) The <TemplateRules> element defines the availability of the property Pset_WallCommon.IsExternal (see Figure 7).
(3) The <Requirements> element (see Figure 6) defines the usage for a set of exchangeRequirements (e.g., the exchangeRequirement with the ID 00000023-0000-0000-0000-000000000352).

This part of ConceptRoot defines the condition under which the requirement test has to be applied to an IFC instance. Some requirements are shown in Table 2. The example shown in Figure 6 defines the applicableRootEntity of IfcWall, which includes all subtypes of IfcWall. Additionally, the <Applicability> element specifies constraints of Pset_WallCommon.IsExternal and Pset_WallCommon.LoadBearing.



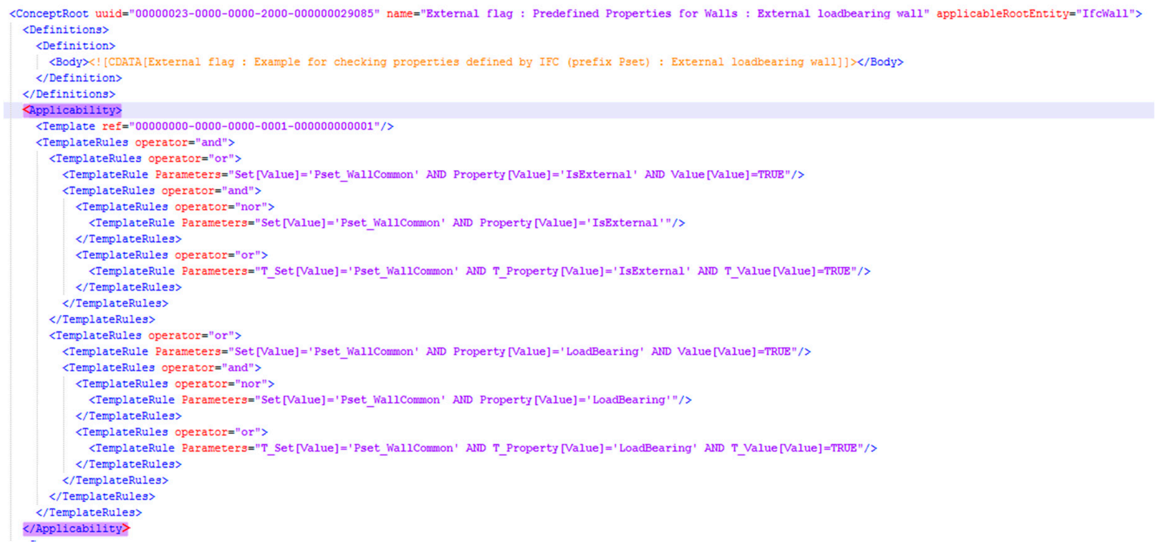
```
</ConceptTemplate>
<ConceptTemplate applicableEntity="IfcObject" applicableSchema="IFC4" name="" uuid="0f7f7621-dc0a-4ab8-8118-64ead2ee3cca">
  <Rules>
    <AttributeRule AttributeName="IsDefinedBy" Cardinality="_asSchema">
      <EntityRules>
        <EntityRule Cardinality="_asSchema" EntityName="IfcRelDefinesByProperties">
          <AttributeRules>
            <AttributeRule AttributeName="RelatingPropertyDefinition" Cardinality="_asSchema">
              <EntityRules>
                <EntityRule Cardinality="_asSchema" EntityName="IfcElementQuantity">
                  <AttributeRules>
                    <AttributeRule AttributeName="Quantities" Cardinality="_asSchema">
                      <EntityRules>
                        <EntityRule Cardinality="_asSchema" EntityName="IfcQuantityVolume">
                          <AttributeRules>
                            <AttributeRule AttributeName="Name" Cardinality="_asSchema" RuleID="IfcQuantityVolumeName"/>
                          </AttributeRules>
                        </EntityRule>
                      </EntityRules>
                    </AttributeRule>
                  </AttributeRules>
                </EntityRule>
              </EntityRules>
            </AttributeRule>
          </AttributeRules>
        </EntityRule>
      </EntityRules>
    </AttributeRule>
  </Rules>
</ConceptTemplate>
```

Figure 5. Example of ConceptTemplate.



```
<ConceptRoot uuid="00000023-0000-0000-2000-000000029085" name="External flag : Predefined Properties for Walls : External loadbearing wall" applicableRootEntity="IfcWall">
  <Definitions>
    <Definition>
      <Body><![CDATA[External flag : Example for checking properties defined by IFC (prefix Pset) : External loadbearing wall]]></Body>
    </Definition>
  </Definitions>
  <Applicability>
  <Concepts>
    <Concept uuid="00000023-0000-0000-0000-000000029085" name="External flag">
      <Definitions>
        <Definition>
          <Body lang="en"><![CDATA[External flag : Example for checking properties defined by IFC (prefix Pset) : External loadbearing wall]]></Body>
        </Definition>
      </Definitions>
      <Template ref="00000000-0000-0000-0001-000000000001"/>
      <Requirements>
        <Requirement applicability="import" exchangeRequirement="00000023-0000-0000-0000-000000000352" requirement="mandatory"/>
      </Requirements>
      <TemplateRules operator="and">
    </Concept>
  </Concepts>
</ConceptRoot>
```

Figure 6. MvdXML snippet.



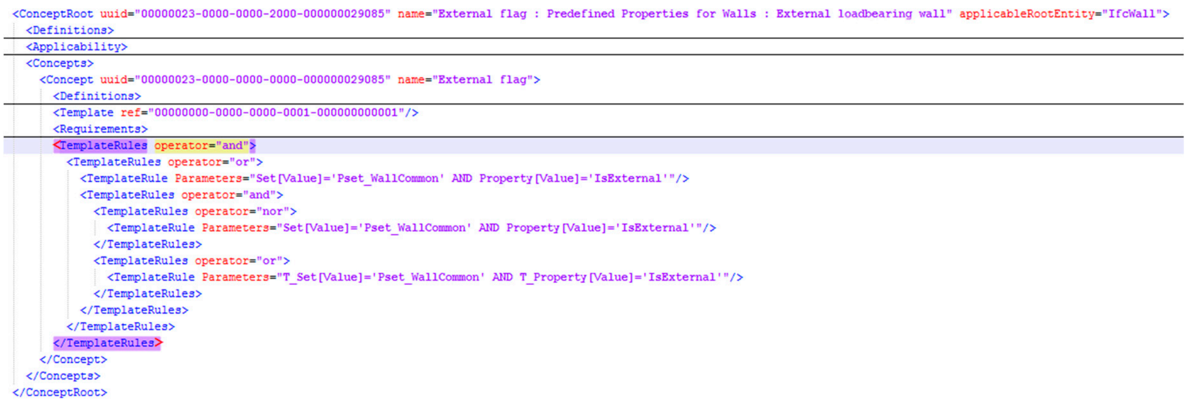
```
<ConceptRoot uuid="00000023-0000-0000-2000-000000029085" name="External flag : Predefined Properties for Walls : External loadbearing wall" applicableRootEntity="IfcWall">
  <Definitions>
    <Definition>
      <Body><![CDATA[External flag : Example for checking properties defined by IFC (prefix Pset) : External loadbearing wall]]></Body>
    </Definition>
  </Definitions>
  <Applicability>
    <Template ref="00000000-0000-0000-0001-000000000001"/>
    <TemplateRules operator="and">
      <TemplateRules operator="or">
        <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='IsExternal' AND Value[Value]=TRUE"/>
        <TemplateRules operator="and">
          <TemplateRules operator="nor">
            <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='IsExternal'"/>
          </TemplateRules>
          <TemplateRules operator="or">
            <TemplateRule Parameters="T_Set[Value]='Pset_WallCommon' AND T_Property[Value]='IsExternal' AND T_Value[Value]=TRUE"/>
          </TemplateRules>
        </TemplateRules>
      </TemplateRules>
      <TemplateRules operator="or">
        <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='LoadBearing' AND Value[Value]=TRUE"/>
        <TemplateRules operator="and">
          <TemplateRules operator="nor">
            <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='LoadBearing'"/>
          </TemplateRules>
          <TemplateRules operator="or">
            <TemplateRule Parameters="T_Set[Value]='Pset_WallCommon' AND T_Property[Value]='LoadBearing' AND T_Value[Value]=TRUE"/>
          </TemplateRules>
        </TemplateRules>
      </TemplateRules>
    </TemplateRules>
  </Applicability>
```

Figure 7. <Applicability> definition of a requirement.

Table 2. Examples of requirements.

Related Concept	Requirements
External loadbearing wall	Show checking results for all ConceptRoots that fulfill applicability test of "external loadbearing wall"
External flag	Test all child node requirements of external flag.
Standard walls	Test for all "standard walls" including "acoustic rating", "external flag", and "load bearing flag"

In Figure 8, the existence of the property Pset_WallCommon.IsExternal is checked. This example is very specific as it tests a property that is already part of the applicability test. If the exchangeRequirement is mandatory, then this test should always pass as it is already required in the applicability test (external walls are identified by this property). If the exchangeRequirement is "excluded", then this test should always fail.



```xml
<ConceptRoot uuid="00000023-0000-0000-2000-000000029085" name="External flag : Predefined Properties for Walls : External loadbearing wall" applicableRootEntity="IfcWall">
  <Definitions>
  <Applicability>
  <Concepts>
    <Concept uuid="00000023-0000-0000-0000-000000029085" name="External flag">
      <Definitions>
      <Template ref="00000000-0000-0000-0001-000000000001"/>
      <Requirements>
      <TemplateRules operator="and">
        <TemplateRules operator="or">
          <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='IsExternal'"/>
          <TemplateRules operator="and">
            <TemplateRules operator="nor">
              <TemplateRule Parameters="Set[Value]='Pset_WallCommon' AND Property[Value]='IsExternal'"/>
            </TemplateRules>
            <TemplateRules operator="or">
              <TemplateRule Parameters="T_Set[Value]='Pset_WallCommon' AND T_Property[Value]='IsExternal'"/>
            </TemplateRules>
          </TemplateRules>
        </TemplateRules>
      </TemplateRules>
    </Concept>
  </Concepts>
</ConceptRoot>
```

Figure 8. <Requirements> definition example.

An mvdXML file contains a broad set of entities and types of ruleset because an mvdXML file contains multiple <ConceptTemplates>. Moreover, each <ConceptTemplate> has multiple referring concepts, which allows integrating a variety of rules and entities in an mvdXML file. After the mvdXML ruleset is completed, the mvdXML checker can check IFC model by rules.

4.1.4. IFC Support Path

The path for IFC support is essential for the creation of a ruleset. The mvdXML rule transformer processes a path like "IfcWindow->IfcObject.IsDefinedBy. IfcRelDefinesByProperties. RelatingPropertyDefinition.IfcPropertySet.HasProperties.IfcPropertySingleValue.Name=IsExternal". The path consists of an applicable object. IFC4 specification requires parameter value. The applicable IfcObject can be an object described in IFC4 or previous schema, for instance, IfcDoor, IfcWindow, IfcWall, or IfcColumn. The rule can easily be applied to a different object by changing the applicable IfcObject, for instance, replacing IfcWindow with IfcDoor.

The elements are separated with operators. The applicable IfcObject is in front of the "->" operator. The part between the "->" operator and "=" operator is specified according to IFC4. Each instance within the IFC4 specification path is separated using a "."; after the "=" operator, the required parameter value is specified.

The second element of the IFC support path is the specification path according to IFC 4. The mvdXML rule transformer is based on IFC4, an example of IFC documentation can be found in Figure 9. The example specifies property sets for IfcObject [49].

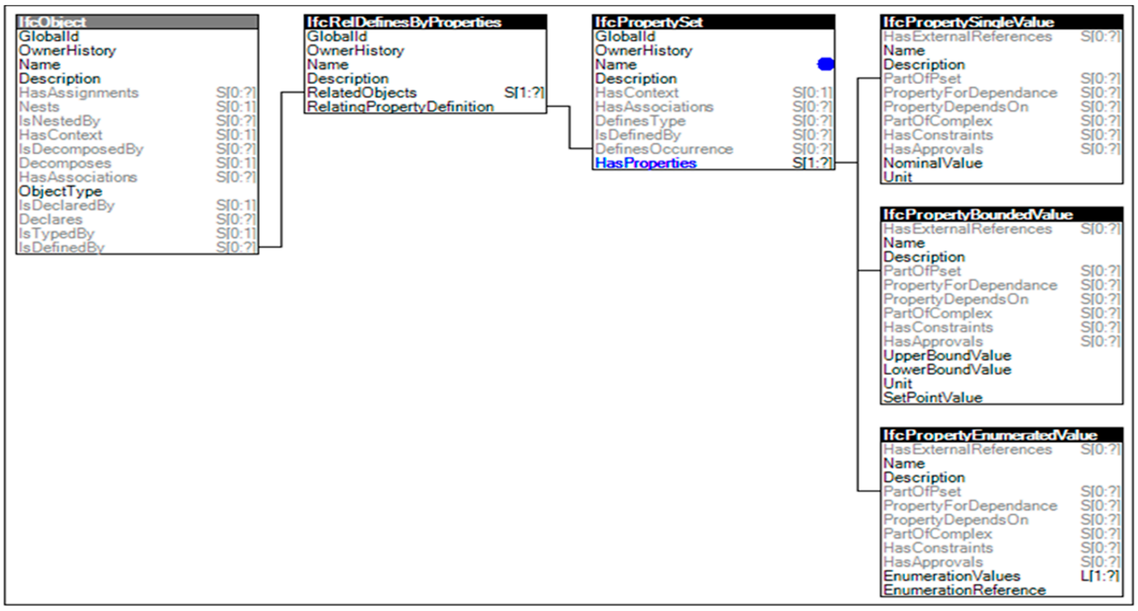


Figure 9. IFC support path.

4.1.5. BCF Report Generator

The IFC model can be checked by the mvdXML ruleset. The IFC objects and attributes from the instance file can be extracted by the developed mvdXML file. Depending on rule types in mvdXML, these values are checked to evaluate their existence, quantity, content, uniqueness, and conditional dependency. The BCF report generator executes the mvdXML checking on the IFC building model (see Figure 10).

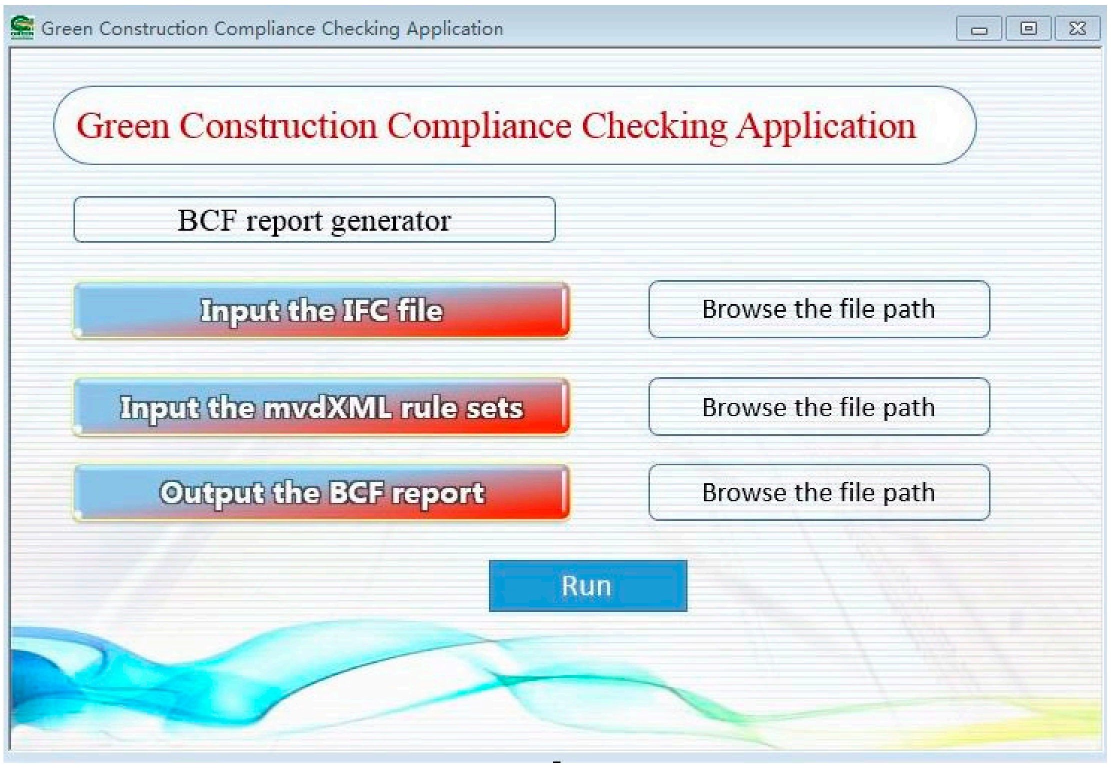

Green Construction Compliance Checking Application
Green Construction Compliance Checking Application
BCF report generator
Input the IFC file
Browse the file path
Input the mvdXML rule sets
Browse the file path
Output the BCF report
Browse the file path
Run

Figure 10. BCF report generator.

4.2. Semantic Technology for Code Checking

4.2.1. Knowledge Base

The knowledge base consists of three types of ontologies, i.e., conceptual model ontology, work item ontology, and construction condition ontology. The purpose of this section is to construct a systematic and computer-readable green construction knowledge base for green construction code checking. A project is composed of a series of work items, and conceptual models can be further classified by work items, which are defined on a project-by-project basis. The conceptual model of green construction is divided into three levels, i.e., basic concept, core concept, and practicability concept. In order to build a better green construction knowledge base, it is necessary to implement inference mapping among conceptual model ontology, work item ontology, and construction condition ontology.

Conceptual Model Ontology

Conceptual model ontology is to extract related concepts from building standards, construction documents, and BIM models, before using the ontology modeling method to build the relationship model of green construction concepts. Through analyzing the relationship between the green construction concepts, the relationship can be described by rules. During green construction code checking, when the information is input to the inference engine, the chosen concepts represent a series of potential requirements. These concepts are regarded as a series of hierarchical concepts, and some semantic relationships exist between concepts. The implementation of inference rules can infer and identify the necessary concepts and their relationships.

Therefore, through the analysis of the characteristics of green construction, the conceptual model ontology includes two parts: (1) structure of concepts; (2) definition of semantic relationships.

(1) Structure of concepts

The first task of conceptual model ontology construction is to define the classification of concept structure and hierarchy structure related to green construction. Based on the previous studies and the characteristics of green construction, this paper defines the relevant attributes of the main classes and their sub-classes including project classification, building product, building feature, etc.

(2) Definition of semantic relationships

When the relationship between concepts is defined, the relationship should be associated with richer semantics. The semantic relationship describes the relationships between concepts in ontology, which can be divided into hierarchical and non-hierarchical [50,51]. The relationship between classes can be divided into internal and external. Through the analysis of the definition of semantic relations, the semantic relations in green construction field are mainly divided into two kinds: hyponymy (also expressed as superclass–subclass) and association. Association can be mainly divided into synonymy relationships (also expressed as Equivalent (x, y)), antonymy relationships (also denoted as Disjoint (x, y)), and meronymy relationships (also expressed as Whole-Part (x, y)). The following is a brief introduction to these relationships:

(1) Equivalent (x, y) is used to describe the similarity between two concepts.
(2) Disjoint (x, y) is used to describe the relationship of independence or antinomy between two concepts.
(3) Whole-Part (x, y) is used to describe the meronymy relationship between two concepts.

Work Item Ontology

A work item is the smallest unit of work defined for building elements in a construction project. The work item can be applied to separate phases from design to operation. The classification of work items is according to national standards for project coding and project naming. A construction process may consist of one or more sub-processes, and a sub-process may consist of a collection of work items for multiple building elements. To establish the ontology of work items, the proposed ontology needs to know the corresponding work items and the corresponding amount of resources necessary. Therefore, the construction of work item ontology can facilitate green construction inspection.

Existing work item ontologies, such as FreeClassOWL ontology, are built based on datasets from the European building and building materials market [52]. By describing the building products and services, they can satisfy a wide range of products, suppliers, warehouses, and other related construction product search needs. More than 88 million triplet business data, 81 product brands, 19 distributors, 56,360 product types, and 1,783,798 products are described in the FreeClassOWL Ontology. Therefore, the work item ontology is huge in terms of scale. During the construction process, participants who have rich knowledge and experience need to be gathered to define the ontology model. The work item ontology proposed in this section is composed of four basic entities: actor, knowledge, resource, and schedule. The basic structure of work item ontology is shown in Figure 11. The actor consists of organization and personnel, the organization is composed of government and company, and personnel is composed of the design, construction, operation, maintenance, and other personnel who assume a variety of roles. Knowledge is generated or updated from each construction process or event. Resource consists of workforce, equipment, material, specification, etc. Construction material is classified according to classification Table 41 of OmniClass. Schedule is the construction schedule, including the actual schedule and planning schedule.

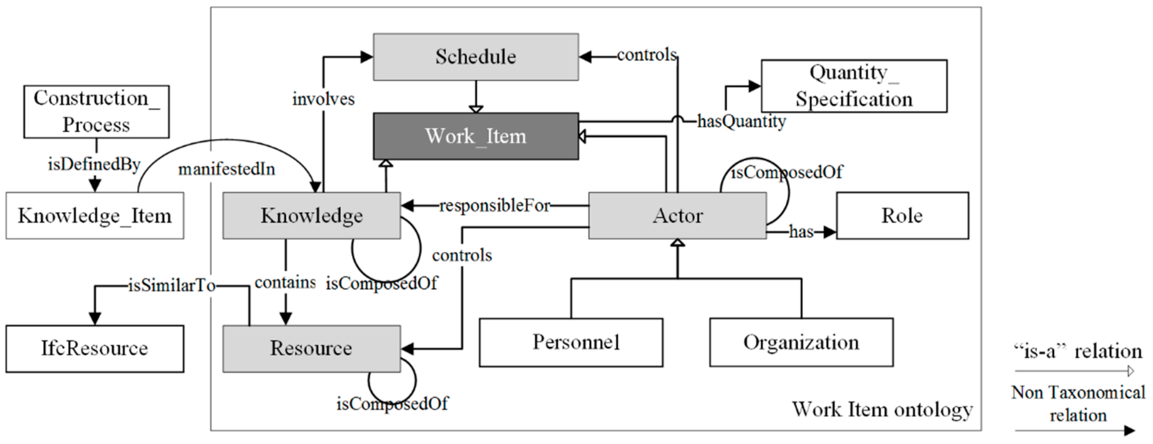
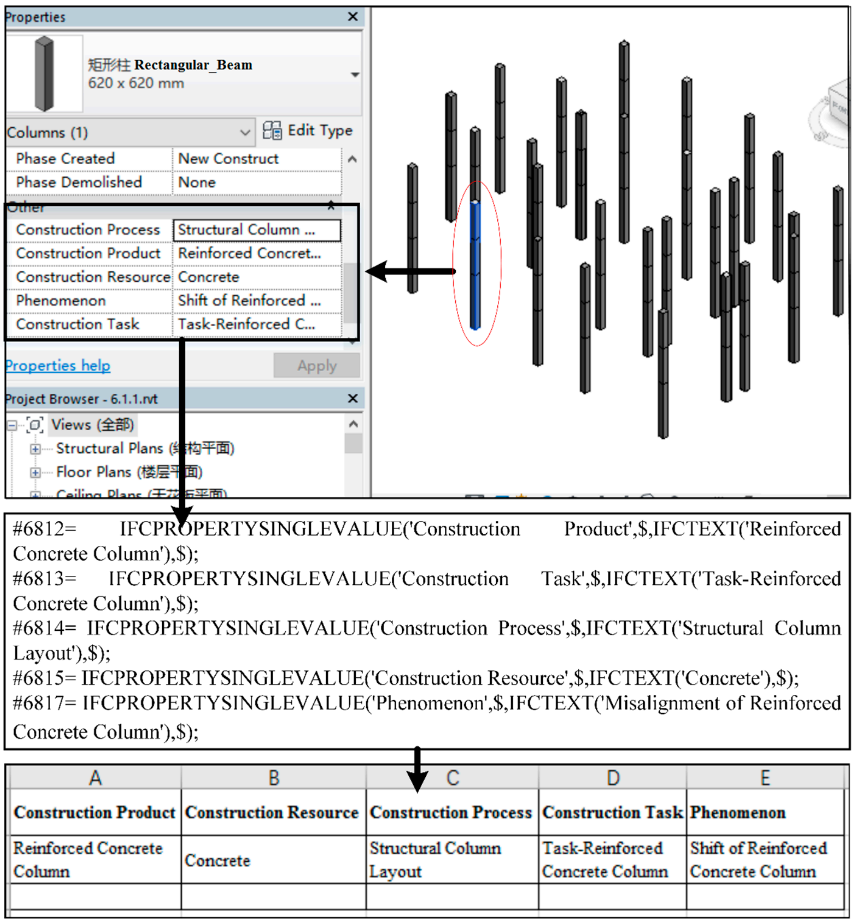


Figure 11. Work item ontology.

Construction Condition Ontology

After building the work item ontology, the product, process, and personnel involved in the ontology can be classified in detail. To realize the information description of the work items about the specific construction conditions, it is very necessary to build mapping relationships between the work item ontology, conceptual model ontology, and construction condition ontology.

Based on the work item ontology and concept model ontology, this paper uses logical inference technology to deduce corresponding concepts or work items through the proposed construction condition ontology. The information in the ontology of construction condition comes from the BIM model (see Figure 12), evaluation criteria, historical data, etc., which include the construction environment, construction machinery, construction sites, construction workers, building models under construction, etc. (see Figure 13). In construction condition ontology, the corresponding work items can be found according to the defined inference rules. For example, specific material information extracted from IFC or a construction document is converted into OWL/RDF format, which can be found in the resource of the construction condition ontology.

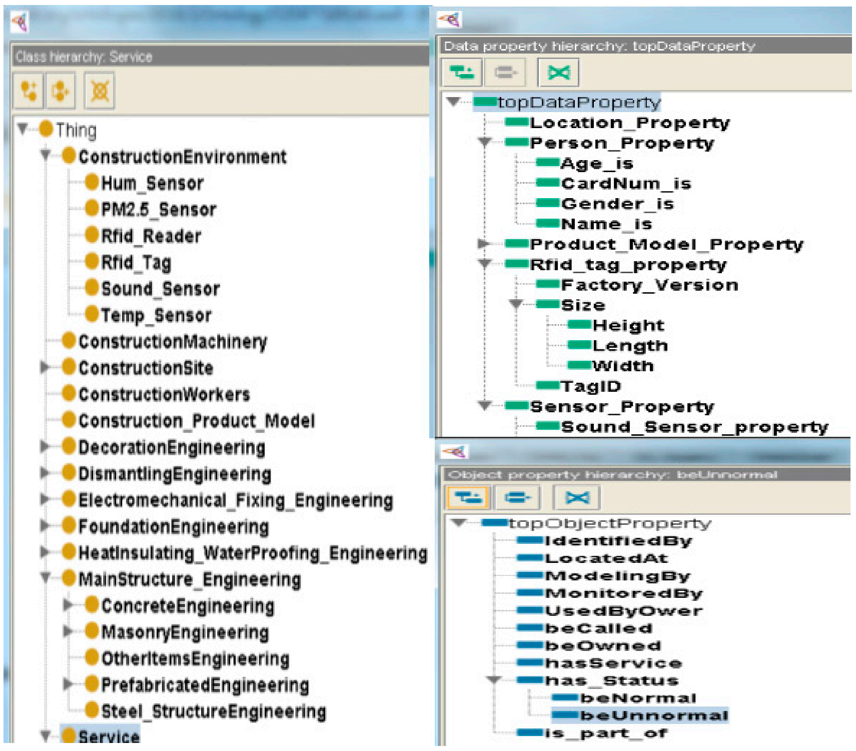


Phase Created	New Construct
Phase Demolished	None
Other	
Construction Process	Structural Column ...
Construction Product	Reinforced Concret...
Construction Resource	Concrete
Phenomenon	Shift of Reinforced ...
Construction Task	Task-Reinforced C...

```
#6812= IFCPROPERTYSINGLEVALUE('Construction Product',$,IFCTEXT('Reinforced
Concrete Column'),$);
#6813= IFCPROPERTYSINGLEVALUE('Construction Task',$,IFCTEXT('Task-Reinforced
Concrete Column'),$);
#6814= IFCPROPERTYSINGLEVALUE('Construction Process',$,IFCTEXT('Structural Column
Layout'),$);
#6815= IFCPROPERTYSINGLEVALUE('Construction Resource',$,IFCTEXT('Concrete'),$);
#6817= IFCPROPERTYSINGLEVALUE('Phenomenon',$,IFCTEXT('Misalignment of Reinforced
Concrete Column'),$);
```

A	B	C	D	E
Construction Product	**Construction Resource**	**Construction Process**	**Construction Task**	**Phenomenon**
Reinforced Concrete Column	Concrete	Structural Column Layout	Task-Reinforced Concrete Column	Shift of Reinforced Concrete Column

Figure 12. Extraction of construction information from BIM model.



```
Class hierarchy: Service
Thing
  ConstructionEnvironment
    Hum_Sensor
    PM2.5_Sensor
    Rfid_Reader
    Rfid_Tag
    Sound_Sensor
    Temp_Sensor
  ConstructionMachinery
  ConstructionSite
  ConstructionWorkers
  Construction_Product_Model
  DecorationEngineering
  DismantlingEngineering
  Electromechanical_Fixing_Engineering
  FoundationEngineering
  HeatInsulating_WaterProofing_Engineering
  MainStructure_Engineering
    ConcreteEngineering
    MasonryEngineering
    OtherItemsEngineering
    PrefabricatedEngineering
    Steel_StructureEngineering
  Service
```

```
Data property hierarchy: topDataProperty
topDataProperty
  Location_Property
  Person_Property
    Age_is
    CardNum_is
    Gender_is
    Name_is
  Product_Model_Property
  Rfid_tag_property
    Factory_Version
    Size
      Height
      Length
      Width
    TagID
  Sensor_Property
    Sound_Sensor_property
```

```
Object property hierarchy: beUnnormal
topObjectProperty
  IdentifiedBy
  LocatedAt
  ModelingBy
  MonitoredBy
  UsedByOwer
  beCalled
  beOwned
  hasService
  has_Status
    beNormal
    beUnnormal
  is_part_of
```

Figure 13. Construction condition ontology.

The needed information to build a green construction knowledge base mainly comes from documents, developers, construction enterprises, governmental agencies, and operation enterprises. Many persons are required to conduct data analysis, extract required information, and build ontology, which takes a lot of manpower and time. Therefore, due to the particularity of green construction code checking, the work of ontology building will be a long-term process.

4.2.2. Semantic Inference

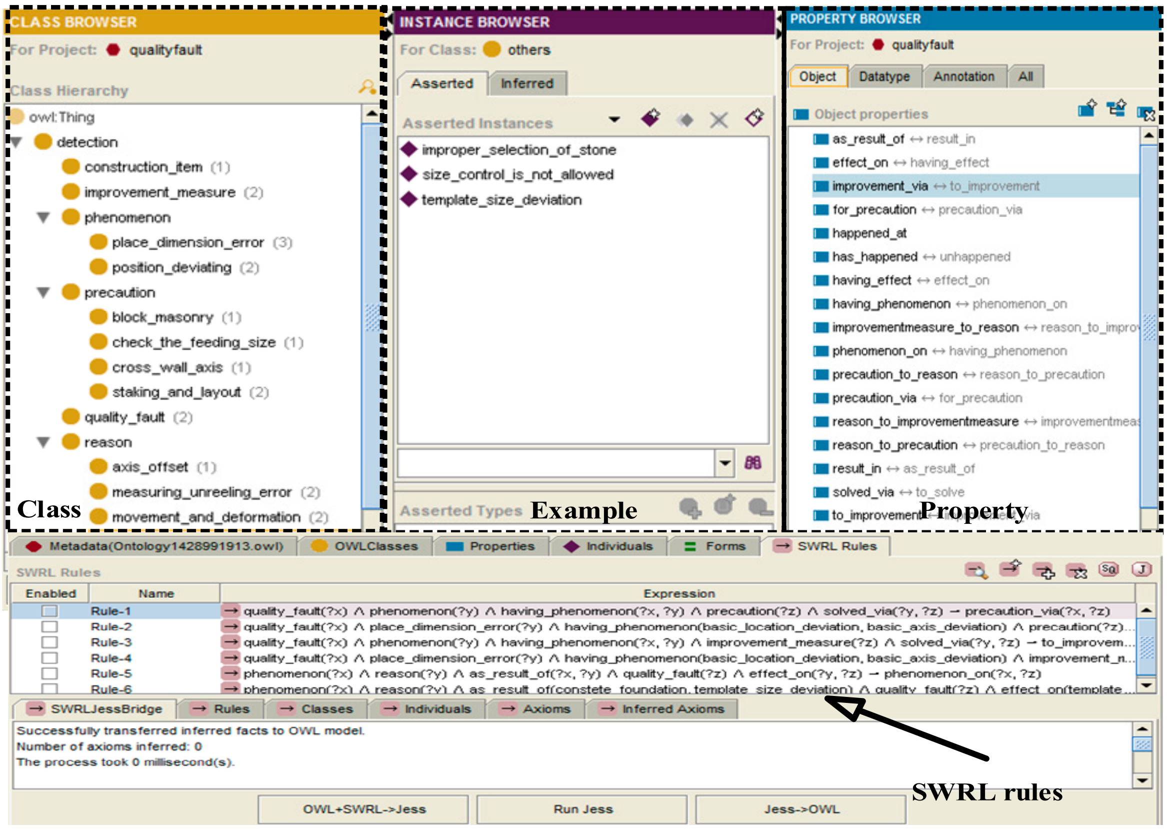
SWRL is used to define related semantic rules during the inference phase; however, SWRL is a language that does not rely on any inference engine and cannot be used without inference tools. Therefore, OWL and SWRL need to be converted into available rules for inference. In the process of ontology inference, JESS is used as an inference engine and consists of a fact base and a rule base. As the most mature inference engine for SWRL, JESS is used in many fact-based systems and is capable of handling ontology-based SWRL rules. In this paper, the JESS rule engine is used to transform OWL and SWRL rules into JESS facts, and it is used to match the rule base and fact base to implement inference. Some specific steps for ontology inference are as follows:

Step 1: In the inference layer, XSLT/XML is used to convert conceptual model ontology, work item ontology, construction condition ontology, and their corresponding inference rules into JESS knowledge for query.

Step 2: The JESS inference engine is used to match the rule base and fact base during the inference process. The inference result is converted to RDF for output using XSLT/XML.

Step 3: Fact and rule bases are stored in the JESS repository.

Some classes and their attribute relationships are defined in Protégé (see Figure 14).



```
owl:Thing
  detection
    construction_item (1)
    improvement_measure (2)
    phenomenon
      place_dimension_error (3)
      position_deviating (2)
    precaution
      block_masonry (1)
      check_the_feeding_size (1)
      cross_wall_axis (1)
      staking_and_layout (2)
    quality_fault (2)
    reason
      axis_offset (1)
      measuring_unreeling_error (2)
      movement_and_deformation (2)
```

```
improper_selection_of_stone
size_control_is_not_allowed
template_size_deviation
```

```
as_result_of ↔ result_in
effect_on ↔ having_effect
improvement_via ↔ to_improvement
for_precaution ↔ precaution_via
happened_at
has_happened ↔ unhappened
having_effect ↔ effect_on
having_phenomenon ↔ phenomenon_on
improvementmeasure_to_reason ↔ reason_to_impro
phenomenon_on ↔ having_phenomenon
precaution_to_reason ↔ reason_to_precaution
precaution_via ↔ for_precaution
reason_to_improvementmeasure ↔ improvementmea
reason_to_precaution ↔ precaution_to_reason
result_in ↔ as_result_of
solved_via ↔ to_solve
to_improvement
```

Enabled	Name	Expression
☐	Rule-1	quality_fault(?x) ∧ phenomenon(?y) ∧ having_phenomenon(?x, ?y) ∧ precaution(?z) ∧ solved_via(?y, ?z) → precaution_via(?x, ?z)
☐	Rule-2	quality_fault(?x) ∧ place_dimension_error(?y) ∧ having_phenomenon(basic_location_deviation, basic_axis_deviation) ∧ precaution(?z)...
☐	Rule-3	quality_fault(?x) ∧ phenomenon(?y) ∧ having_phenomenon(?x, ?y) ∧ improvement_measure(?z) ∧ solved_via(?y, ?z) → to_improvem...
☐	Rule-4	quality_fault(?x) ∧ place_dimension_error(?y) ∧ having_phenomenon(basic_location_deviation, basic_axis_deviation) ∧ improvement_n...
☐	Rule-5	phenomenon(?x) ∧ reason(?y) ∧ as_result_of(?x, ?y) ∧ quality_fault(?z) ∧ effect_on(?y, ?z) → phenomenon_on(?x, ?z)
☐	Rule-6	phenomenon(?x) ∧ reason(?y) ∧ as_result_of(constete_foundation, template_size_deviation) ∧ quality_fault(?z) ∧ effect_on(template...

```
Successfully transferred inferred facts to OWL model.
Number of axioms inferred: 0
The process took 0 millisecond(s).
```

Figure 14. Some classes and instances built by Protégé.

The JESS inference engine can only interpret the JESS code; thus, the structured presentation languages OWL and SWRL need to be transformed into the JESS-encoded form using the OWL2Jess and SWRL2Jess translators. Both translators are built-in plug-ins of Protégé to integrate heterogeneous information between OWL, SWRL, and JESS. IFC files can be converted from models saved in ISO STEP (Standard for the Exchange of Product model data). The models are specified in EXPRESS and represented by OWL. EXPRESS is a standard data modeling language for product data. Table 3 shows the data type mapping relationships between EXPRESS, OWL, and JESS.

Table 3. The data type mapping relationships between EXPRESS, OWL, and JESS.

EXPRESS	OWL	JESS
real, integer, number, etc.	owl: real, xsd: integer, xsd: decimal, owl: rational, etc	jess.ru.integer, jess.ru.float, jess.ru.long
string	xsd: string	jess.ru.string
boolean	xsd: boolean	The atoms “TRUE” and “FALSE”

The constructions of conceptual model ontology, work item ontology, and construction condition ontology are beneficial for the green construction knowledge sharing and reuse. OWL can express the concept of knowledge of green construction. SWRL can describe inference rules in ontologies. Through the combination of SWRL and the JESS inference engine, inference results of green construction code checking can be easily obtained.

5. Case Study of Green Construction Code Checking

In this paper, some types of rules and the corresponding clauses in GB/T50905-2014 are shown as examples.

Class 1: Code checking based on simple defined attribute value in IFC schema.

9.3.5 Extra item: staff dormitory must meet the requirements of at least two square meters per person.

mvdXML rule:

IfcPhysicalQuantity.IfcPhysicalSimpleQuantity.IfcQuantityArea.AreaValue>2*4 (there are four people in every room in this case).

SQWRL rule:

```
Room(?r) ˆ hasWidth(?r, ?width) ˆ hasHeight(?r, ?height) ˆ
    swrlm:eval(?area, “width * height”, ?width, ?height)
    -> sqwrl:select(?area).
```

Class 2: Code checking based on the extension of IFC entities.

6.2.1 General item: fly ash, slag admixture, and other new materials should be used during the construction period.

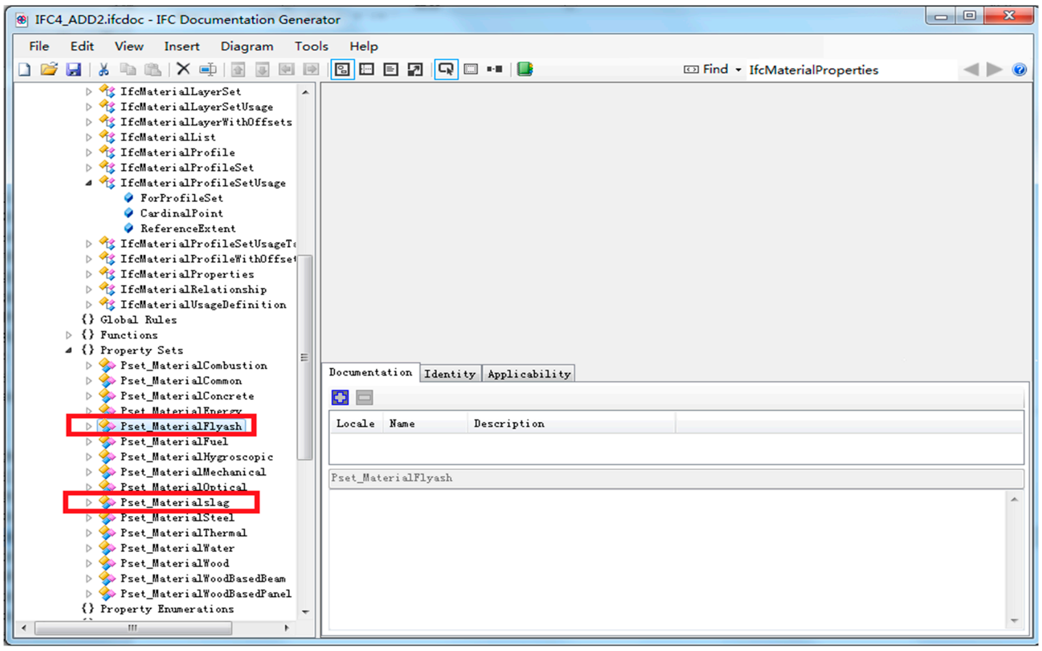
The fly ash and slag admixture must be inserted into IFC baseline, because they are not defined in existing IFC (see Figure 15).

mvdXML rule:

IfcObject.IsDefinedBy.IfcRelDefinesByProperties.RelatingPropertyDefinition.IfcMaterialDefinition. HasProperties.IfcMaterialProperties.Name=Flyash

IfcObject.HasAssocialtions.IfcRelAssociatesMaterial.RelatingMaterial.IfcMaterial.Material. Name= slag.

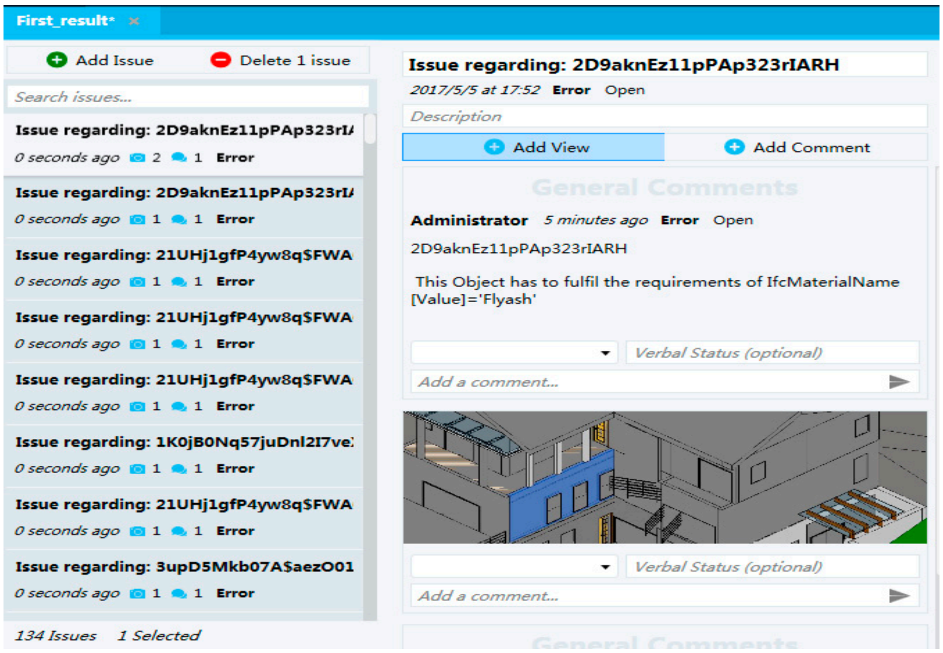
After the checking is executed, the mvdXML checker captures each generated issue in a BCF report which mainly includes a markup file and a viewpoint file. The markup file is where most of the information about an issue resides. In a markup file, information about the model, about the issue, and about each comment on the issue can be obtained. The generated issue comments in the markup file contain the description of the "concept" defined in the mvdXML file to make domain users understand the requirement that is violated. The viewpoint file contains all the information about the camera, the elements in the view, and the software used. BIM software can be used to find and analyze the generated issues from the mvdXML checker. The BCF report can be opened with BCFier in Revit (see Figure 16).



IfcMaterialLayerSet
IfcMaterialLayerSetUsage
IfcMaterialLayerWithOffsets
IfcMaterialList
IfcMaterialProfile
IfcMaterialProfileSet
IfcMaterialProfileSetUsage
ForProfileSet
CardinalPoint
ReferenceExtent
IfcMaterialProfileSetUsageT
IfcMaterialProfileWithOffset
IfcMaterialProperties
IfcMaterialRelationship
IfcMaterialUsageDefinition
Global Rules
Functions
Property Sets
Pset_MaterialCombustion
Pset_MaterialCommon
Pset_MaterialConcrete
Pset_MaterialEnergy
Pset_MaterialFlyash
Pset_MaterialFuel
Pset_MaterialHygroscopic
Pset_MaterialMechanical
Pset_MaterialOptical
Pset_Materialslag
Pset_MaterialSteel
Pset_MaterialThermal
Pset_MaterialWater
Pset_MaterialWood
Pset_MaterialWoodBasedBeam
Pset_MaterialWoodBasedPanel
Property Enumerations

Pset_MaterialFlyash

Figure 15. Insert the new properties in IFC.



Issue regarding: 2D9aknEz11pPAp323rI/
0 seconds ago 2 1 Error

Issue regarding: 2D9aknEz11pPAp323rI/
0 seconds ago 1 1 Error

Issue regarding: 21UHj1gfP4yw8q$FWA
0 seconds ago 1 1 Error

Issue regarding: 21UHj1gfP4yw8q$FWA
0 seconds ago 1 1 Error

Issue regarding: 21UHj1gfP4yw8q$FWA
0 seconds ago 1 1 Error

Issue regarding: 1K0jB0Nq57juDnI2I7ve
0 seconds ago 1 1 Error

Issue regarding: 21UHj1gfP4yw8q$FWA
0 seconds ago 1 1 Error

Issue regarding: 3upD5Mkb07A$aezO01
0 seconds ago 1 1 Error

134 Issues 1 Selected

Issue regarding: 2D9aknEz11pPAp323rIARH
2017/5/5 at 17:52 Error Open

Administrator 5 minutes ago Error Open
2D9aknEz11pPAp323rIARH
This Object has to fulfil the requirements of IfcMaterialName [Value]='Flyash'

Figure 16. The BCF report of the code checking.

Class 3: Code checking based on simple conceptual reasoning.

9.2.4 Extra item: Cast-in-place concrete raw materials should be mixed in construction site.

The constructed ontology model and rules are stored in the knowledge base and rule base, respectively. They are integrated to conduct inference in the JESS inference part. The inference result from the JESS inference engine is shown in Figure 17. The result shows that "cast-in-place concrete" is composed of materials "cement", "sand", "stone", and "water". The "Rectangular_Beam" and its attributes are defined in the conceptual model ontology, while "cement", "sand", "stone", and "water" are defined in the work item ontology. The inference process is related to construction condition knowledge. Therefore, the realization of this inference process is based on the concept model ontology, work item ontology, and construction condition ontology.

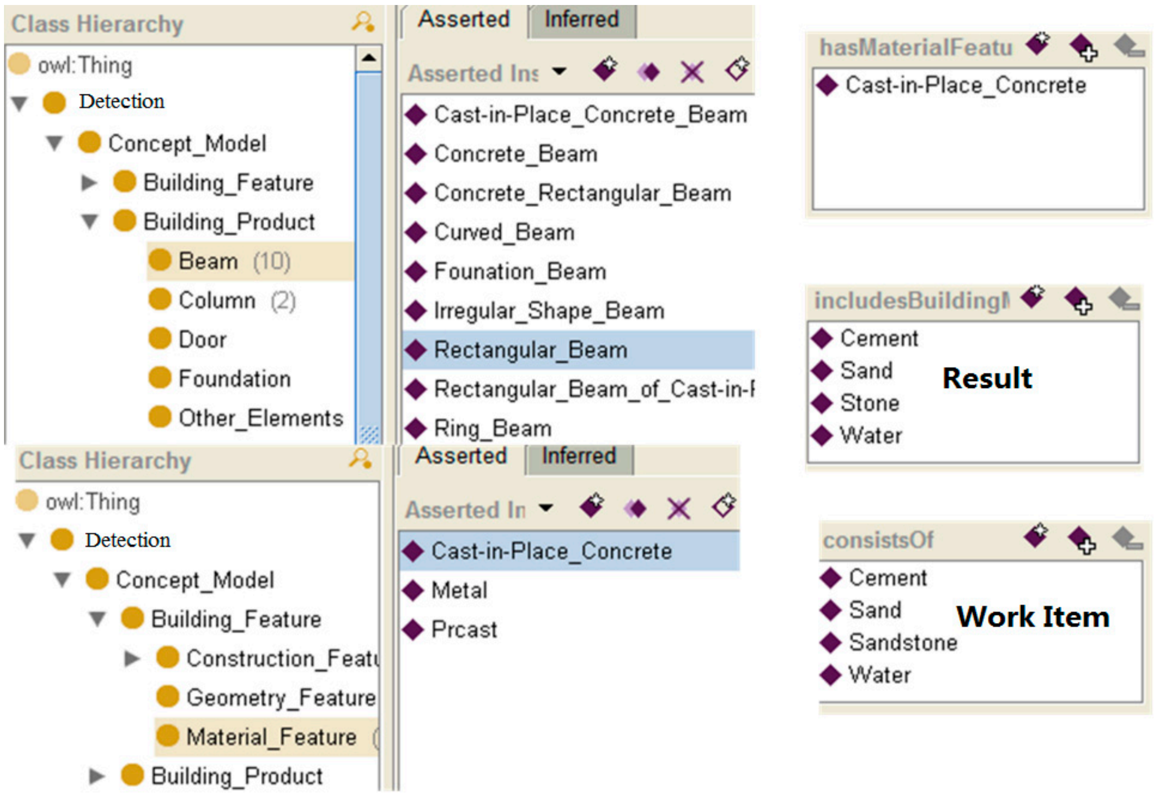


Figure 17. Inference result.

SQWRL rule:

Cast_In_Place_Concrete(?c) ˆ hasComponents(?c, ?z) ˆ hasMaterials(?z, ?d) -> sqwrl:select(?d) ˆ sqwrl:orderBy(?d).

Class 4: Code checking based on construction condition reasoning.

3.3.2 Extra item: Construction field noise emission at daylight should not exceed 70 dB (A) and should not exceed 55 dB at night (A).

This kind of clause is closely related to the construction site environment. The noise monitoring in the construction process is embodied in the construction condition ontology. The PM2.5_Sensor and Sound_Sensor are set in ontology (see Figure 18). The diagram of classes and properties in construction condition ontology is shown in Figure 19.

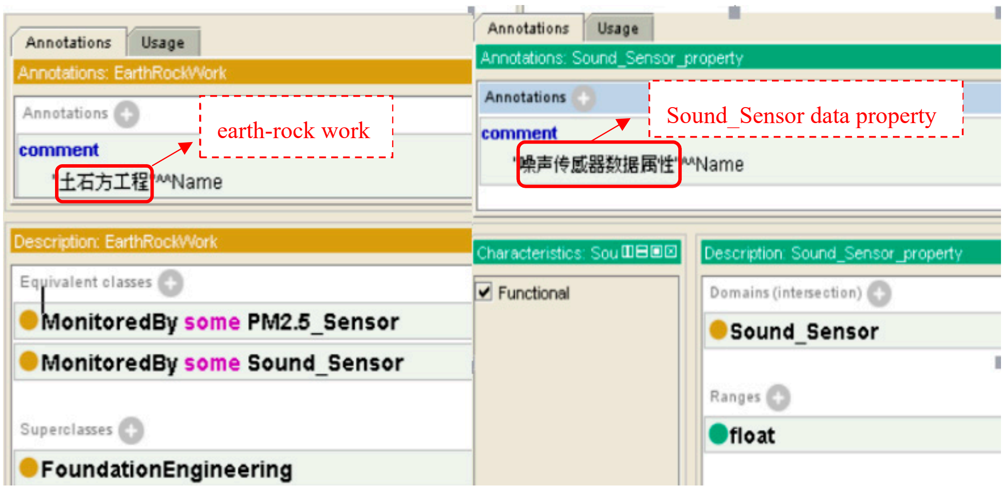

Annotations
Usage
Annotations: EarthRockWork
comment
土石方工程^^Name
earth-rock work
Description: EarthRockWork
Equivalent classes
MonitoredBy some PM2.5_Sensor
MonitoredBy some Sound_Sensor
Superclasses
FoundationEngineering
Annotations: Sound_Sensor_property
噪声传感器数据属性^^Name
Sound_Sensor data property
Characteristics: Sou
Functional
Description: Sound_Sensor_property
Domains (intersection)
Sound_Sensor
Ranges
float

Figure 18. The sensor set in construction condition ontology.

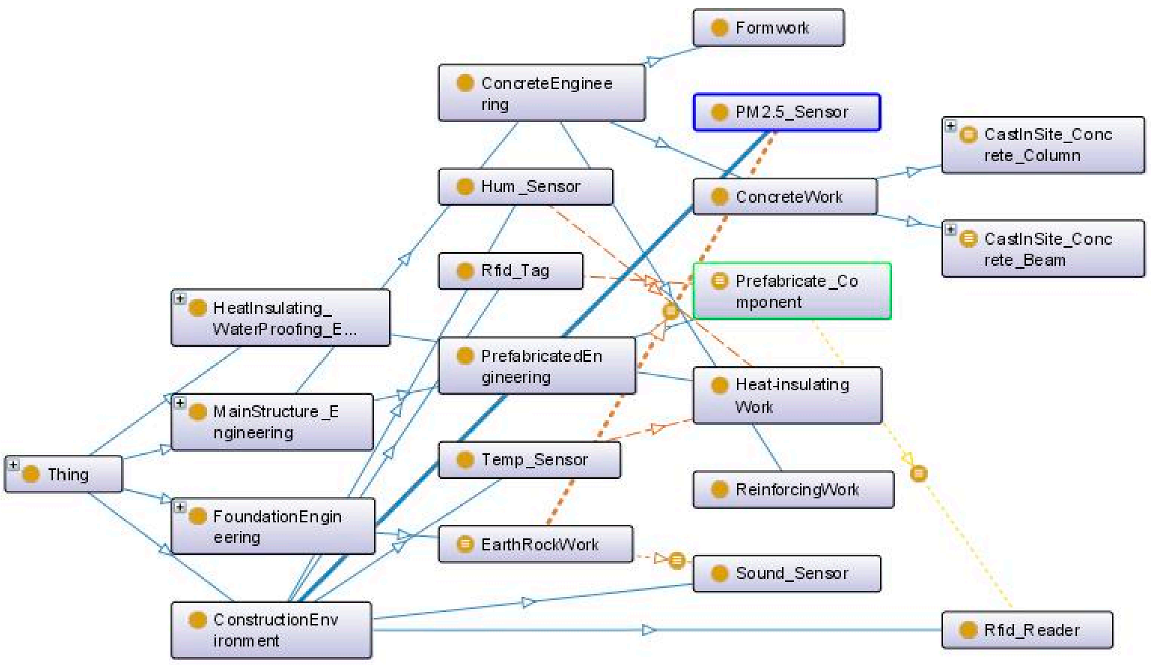


Figure 19. The diagram of classes and properties in construction condition ontology.

For example, if the entry place of the construction site is the fixed position of the Radio-frequency identification (RFID) reader, the component, material, and the person after passing the RFID tag will have the position information, which can be used for ontology inference, along with other attributes, to judge the execution of the service. The construction site will be equipped with noise sensors to collect noise data; when the noise data exceeds the threshold specified in green construction code, an early warning will be triggered automatically. The overrun alarm for noise values is shown in Figure 20. The rule expressions are as follows:

SWRL rule1

(?Rfid_Tag t1:beOwnedBy ?Prefabricate_Component)
(?Rfid_Tag t1:IdentifiedBy ?entrance_Rfid_Reader)
-> (?Prefabricate_Component t1:locatedAt ?Entrance).

SWRL rule2

```
        (?Prefabricate_Component t1:locatedAt ?Entrance)
        ->  (?Service t1:beCalled t1:Register).
```

SWRL rule3

```
            (EarthRockWork t1:MoniterdBy ?Sound_Sensor)
            (?Sound_Sensor t1:hasStatus t1:is_unnormal)
        ->  (?Service t1:beCalled t1:Alarm).
```

SQWRL rule

```
        Sound_Sensor (?p) ˆ hasValue(?p, ?s) ˆ swrlb:greaterThan(?s, 70) -> unnormal(?p).
```

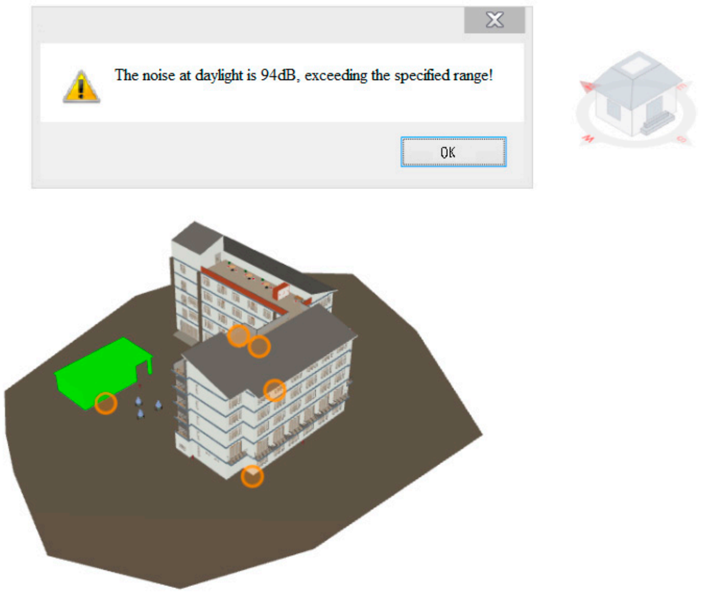
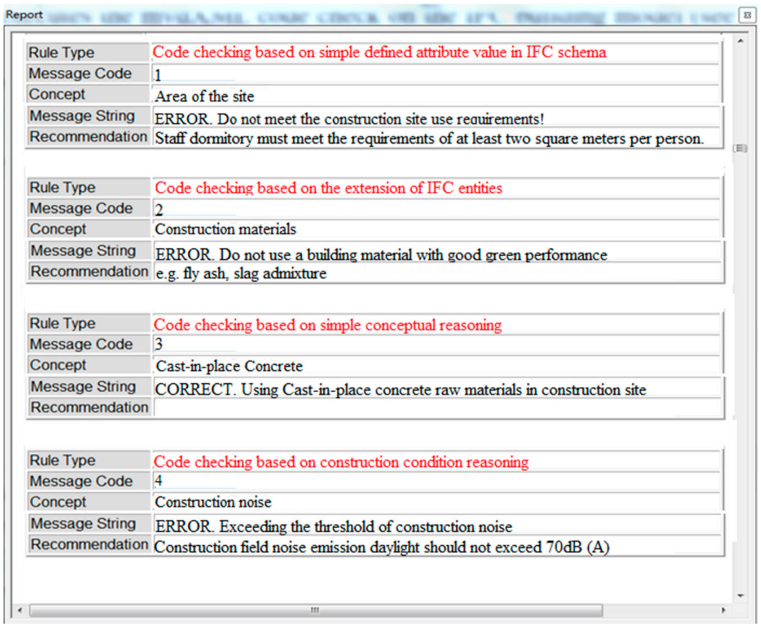

The noise at daylight is 94dB, exceeding the specified range!

Figure 20. The overrun alarm for noise values.

Finally, all the code checking results can be summarized to a HTML report (see Figure 21).



Rule Type	Code checking based on simple defined attribute value in IFC schema
Message Code	1
Concept	Area of the site
Message String	ERROR. Do not meet the construction site use requirements!
Recommendation	Staff dormitory must meet the requirements of at least two square meters per person.

Rule Type	Code checking based on the extension of IFC entities
Message Code	2
Concept	Construction materials
Message String	ERROR. Do not use a building material with good green performance
Recommendation	e.g. fly ash, slag admixture

Rule Type	Code checking based on simple conceptual reasoning
Message Code	3
Concept	Cast-in-place Concrete
Message String	CORRECT. Using Cast-in-place concrete raw materials in construction site
Recommendation	

Rule Type	Code checking based on construction condition reasoning
Message Code	4
Concept	Construction noise
Message String	ERROR. Exceeding the threshold of construction noise
Recommendation	Construction field noise emission daylight should not exceed 70dB (A)

Figure 21. The HTML report of code checking.

6. Conclusions and Perspectives

Although professionals in the AEC domain realize the importance of green construction, there is no efficient and accurate evaluation method for green construction. Traditional green construction evaluation is usually completed by manual inspection, with low efficiency and high error rate. Therefore, construction industry practitioners put forward urgent requirements for a more efficient and convenient approach to conduct green construction code checking.

Based on the particularity of green construction standards and the limitation of existing code-checking approaches, this paper combines mvdXML technology and semantic web technology to organize, store, and re-use green construction knowledge. In our approach, BIM, logical rule, ontology, and semantic web query language are integrated to conduct green construction code checking. These techniques are used for automated information transformation and extension, knowledge base construction, logical inference, and information query. The automation is facilitated by semantic-based and logic-based representations, which is general and flexible. The approach is composed of five main parts: data source, knowledge base, rule base, JESS inference, and query.

In this paper, code checking of green construction standards was classified into four classes based on the difficulty level to meet the requirements of the clauses, and the particularity of green construction code checking was analyzed. Based on the above classification and analysis, several types of green construction rules were checked. The proposed BIM-based code-checking approach for green construction was validated by a case study.

The proposed approach not only focuses on quantitative requirements, but also supports the checking of other types of requirements, such as checking based on semantic information of BIM and checking based on project criteria, owner requirement, etc.

7. Limitations and Future Work

The proposed approach offers a solution to code checking for green construction. However, there are some limitations in this approach, and there is still some work to be done in future research. Firstly, the scoring rules of existing green construction standards are relatively rough, which results in weak operability; thus, a more detailed standard is expected to be issued to improve green construction inspection. Secondly, although the approach proposed in this paper enables making various rules, parameters that are not predefined cannot be checked without being encoded into programming language. Thirdly, automated regulatory code checking requires automated extraction of requirements from regulatory textual documents, and then the extracted requirements need to be transformed into a formalized format that enables JESS inference, which still requires human judgement and cannot reach the level of artificial intelligence. Information extraction (IE) in code checking is a challenging task that requires complex analysis and processing of text. Some researchers started using deep learning algorithms like NLP (Natural Language Processing) to solve this kind of problem, but it is still imperfect. The field of automatic code checking is emerging, and it will continue to offer more help in the future.

Author Contributions: S.J. conceived the idea for the paper, oversaw its development and supervised all aspects of this research. Z.W. designed the methodology of code checking. B.Z. analyzed and classified code checking of green construction. H.S.C. conducted the case study and analyzed the results. All authors have reviewed and approved the final manuscript.

Funding: This research was funded by the National Key R&D Program of China grant number 2016YFC0702107.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cole, R.J. Energy and greenhouse gas emissions associated with the construction of alternative structural systems. *Build. Environ.* **1998**, *34*, 335–348. [CrossRef]
2. Wong, J.K.W.; Zhou, J. Enhancing environmental sustainability over building life cycles through green BIM: A review. *Autom. Constr.* **2015**, *57*, 156–165. [CrossRef]

3. González, M.J.; García Navarro, J. Assessment of the decrease of CO2 emissions in the construction field through the selection of materials: Practical case study of three houses of low environmental impact. *Build. Environ.* **2006**, *41*, 902–909. [CrossRef]
4. Won, J.; Cheng, J.C.P.; Lee, G. Quantification of construction waste prevented by BIM-based design validation: Case studies in South Korea. *Waste Manag.* **2016**, *49*, 170–180. [CrossRef] [PubMed]
5. IFC Model View Definition Format. Available online: http://www.buildingsmart-tech.org/downloads/accompanying-documents/formats/mvdxml-documentation/MVD_Format_V2_Proposal_080128.pdf (accessed on 17 December 2018).
6. Connor, M.O.; Das, A. SQWRL: A query language for OWL. In Proceedings of the International Conference on Owl: Experiences and Directions, Chantilly, VA, USA, 23–24 October 2009; pp. 208–215.
7. Fudholi, D.H.; Maneerat, N.; Varakulsiripunth, R.; Kato, Y. Application of protégé, SWRL and SQWRL in fuzzy ontology-based menu recommendation. In Proceedings of the ISPACS 2009—2009 International Symposium on Intelligent Signal Processing and Communication Systems, Kanazawa, Japan, 7–9 December 2009; pp. 631–634.
8. BIM Levels Explained. Available online: https://www.thenbs.com/knowledge/bim-levels-explained (accessed on 19 November 2018).
9. Ismail, A.S.; Ali, K.N.; Iahad, N.A. A Review on BIM-based automated code compliance checking system. In Proceedings of the IEEE International Conference on Research and Innovation in Information Systems, ICRIIS, Langkawi, Malaysia, 16–17 July 2017.
10. Solihin, W.; Eastman, C. Classification of rules for automated BIM rule checking development. *Autom. Constr.* **2015**, *53*, 69–82. [CrossRef]
11. Eastman, C.; Lee, J.; Jeong, Y.; Lee, J. Automatic rule-based checking of building designs. *Autom. Constr.* **2009**, *18*, 1011–1033. [CrossRef]
12. Dimyadi, J.; Solihin, W.; Hjelseth, E. Classification of BIM-based Model checking concepts. *J. Inf. Technol. Constr.* **2016**, *21*, 354–370.
13. Krijnen, T.; Berlo, L.A.H.M. Van Methodologies for requirement checking on building models: A technology overview. In Proceedings of the the 13th International Conference on Design and Decision Support Systems in Architecture and Urban Planning, Eindhoven, The Netherlands, 27–28 June 2016; pp. 1–11.
14. Fahad, M.; Bus, N.; Andrieux, F. Towards Mapping Certification Rules over BIM Towards Mapping Certification Rules over BIM. In Proceedings of the the 33rd CIB W78 Conference, Brisbane, Australia, 31 October–2 November 2016.
15. Choi, J.; Kim, I. An Approach to Share Architectural Drawing Information and Document Information for Automated Code Checking System. *Tsinghua Sci. Technol.* **2008**, *13*, 171–178. [CrossRef]
16. Lee, Y.C.; Eastman, C.M.; Lee, J.K. Validations for ensuring the interoperability of data exchange of a building information model. *Autom. Constr.* **2015**, *58*, 176–195. [CrossRef]
17. Nawari, N. The Challenge of Computerizing Building Codes in a BIM Environment. *Comput. Civ. Eng.* **2012**, *1*, 285–292.
18. Nawari, N. SmartCodes and BIM. In Proceedings of the Structures Congress 2013, Pittsburgh, PA, USA, 2–4 May 2013; pp. 928–937.
19. Ilhan, B.; Yaman, H. Green building assessment tool (GBAT) for integrated BIM-based design decisions. *Autom. Constr.* **2016**, *70*, 26–37. [CrossRef]
20. Ciribini, A.L.C.; Mastrolembo Ventura, S.; Paneroni, M. Implementation of an interoperable process to optimise design and construction phases of a residential building: A BIM Pilot Project. *Autom. Constr.* **2016**, *71*, 62–73. [CrossRef]
21. Zhang, S.; Teizer, J.; Lee, J.K.; Eastman, C.M.; Venugopal, M. Building Information Modeling (BIM) and Safety: Automatic Safety Checking of Construction Models and Schedules. *Autom. Constr.* **2013**, *29*, 183–195. [CrossRef]
22. Kasim, T.; Li, H.; Rezgui, Y.; Beach, T. Automated Sustainability Compliance Checking Process: Proof of Concept. In Proceedings of the 13th International Conference on Construction Applications of Virtual Reality (ConVR), London, UK, 30–31 October 2013; pp. 11–21.
23. Lee, Y.C.; Eastman, C.M.; Solihin, W.; See, R. Modularized rule-based validation of a BIM model pertaining to model views. *Autom. Constr.* **2016**, *63*, 1–11. [CrossRef]

24. Dimyadi, J.; Solihin, W.; Preidel, C.; Borrmann, A. Towards code compliance checking on the basis of a visual programming language. *J. Inf. Technol. Constr.* **2016**, *21*, 402–421.
25. Dimyadi, J.; Solihin, W.; Solihin, W.; Eastman, C. A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. *J. Inf. Technol. Constr.* **2016**, *21*, 370–402.
26. Zhang, J.; El-Gohary, N.M. Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *J. Comput. Civ. Eng.* **2016**, *30*, 04015014. [CrossRef]
27. Zhang, J.; El-Gohary, N.M. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Autom. Constr.* **2017**, *73*, 45–57. [CrossRef]
28. Zhang, J.; El-Gohary, N.M. Automated Information Transformation for Automated Regulatory Compliance Checking in Construction. *J. Comput. Civ. Eng.* **2015**, *29*, B4015001. [CrossRef]
29. Jiang, L.; Leicht, R.M. Supporting automated Constructability checking for formwork construction: An ontology. *J. Inf. Technol. Constr.* **2016**, *21*, 456–478.
30. Pauwels, P.; Terkaj, W. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Autom. Constr.* **2016**, *63*, 100–133. [CrossRef]
31. Pauwels, P.; de Farias, T.M.; Zhang, C.; Roxin, A.; Beetz, J.; De Roo, J.; Nicolle, C. A performance benchmark over semantic rule checking approaches in construction industry. *Adv. Eng. Inform.* **2017**, *33*, 68–88. [CrossRef]
32. RDF 1.1 Semantics. Available online: https://www.w3.org/TR/rdf11-mt/ (accessed on 17 December 2018).
33. RDF 1.1 XML Syntax. Available online: https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/ (accessed on 17 December 2018).
34. OWL 2 Web Ontology Language Document Overview (Second Edition). Available online: https://www.w3.org/TR/owl2-overview/ (accessed on 18 December 2018).
35. Bouzidi, K.R.; Fies, B.; Faron-Zucker, C.; Zarli, A.; Thanh, N. Le Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry. *Future Internet* **2012**, *4*, 830–851. [CrossRef]
36. Do, S.L.; Shin, M.; Baltazar, J.C.; Kim, J. Energy benefits from semi-transparent BIPV window and daylight-dimming systems for IECC code-compliance residential buildings in hot and humid climates. *Solar Energy* **2017**, *155*, 291–303. [CrossRef]
37. Bakillah, M.; Mostafavi, M.A.; Liang, S.H.L. Enriching SQWRL Queries in Support of Geospatial Data Retrieval from Multiple and Complementary Sources. In *Proceedings of the Sixth International Workshop on Semantic and Conceptual Issues in GIS*; Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L., Eds.; Springer: Florence, Italy, 2012; pp. 241–250.
38. Zhong, B.T.; Ding, L.Y.; Luo, H.B.; Zhou, Y.; Hu, Y.Z.; Hu, H.M. Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking. *Autom. Constr.* **2012**, *28*, 58–70. [CrossRef]
39. Lu, Y.; Li, Q.; Zhou, Z.; Deng, Y. Ontology-based knowledge modeling for automated construction safety checking. *Saf. Sci.* **2015**, *79*, 11–18. [CrossRef]
40. Ding, L.Y.; Zhong, B.T.; Wu, S.; Luo, H.B. Construction risk knowledge management in BIM using ontology and semantic web technology. *Saf. Sci.* **2016**, *87*, 202–213. [CrossRef]
41. Lee, Y.C.; Eastman, C.M.; Solihin, W. An ontology-based approach for developing data exchange requirements and model views of building information modeling. *Adv. Eng. Inform.* **2016**, *30*, 354–367. [CrossRef]
42. Zhang, H.; Zhao, W.; Gu, J.; Liu, H.; Gu, M. Semantic Web Based Rule Checking of Real-World Scale BIM Models: A Pragmatic Method. *Semant. Web*. 2017. Available online: http://www.semantic-web-journal.net/system/files/swj1621.pdf (accessed on 29 March 2019).
43. de Farias, T.M.; Roxin, A.; Nicolle, C. A rule-based methodology to extract building model views. *Autom. Constr.* **2018**, *92*, 214–229. [CrossRef]
44. Schwabe, K.; König, M.; Teizer, J. BIM Applications of Rule-based Checking in Construction Site Layout Planning Tasks. In Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC), Auburn, AL, USA, 18–21 July 2016; pp. 209–217.
45. mvdXML: Specification of a Standardized Format to Define and Exchange Model View Definitions with Exchange Requirements and Validation Rules. Available online: http://www.buildingsmart-tech.org/downloads/accompanying-documents/formats/mvdxml-documentation/mvdXML_V1-0.pdf (accessed on 18 December 2018).

46. Van Berlo, L.; Krijnen, T. Using the BIM Collaboration Format in a Server Based Workflow. *Procedia Environ. Sci.* **2014**, *22*, 325–332. [CrossRef]
47. Zhang, C.; Beetz, J.; Weise, M. Interoperable validation for IFC building models using open standards. *J. Inf. Technol. Constr.* **2015**, *20*, 24–39. [CrossRef]
48. IFC Model Checking Based on mvdXML 1.1. Available online: https://core.ac.uk/display/80693771 (accessed on 18 December 2018).
49. Industry Foundation Classes—Version 4—Addendum 2. Available online: http://www.buildingsmart-tech.org/ifc/IFC4/Add2/html/ (accessed on 18 December 2018).
50. Khan, S.; Safyan, M. Semantic matching in hierarchical ontologies. *J. King Saud Univ. Comput. Inf. Sci.* **2014**, *26*, 247–257. [CrossRef]
51. Punuru, J.; Chen, J. Extraction of Non-hierarchical Relations from Domain Texts. In Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining, Honolulu, HI, USA, 1–5 April 2007; pp. 444–449.
52. Radinger, A.; Rodriguez-Castro, B.; Stolz, A.; Hepp, M. BauDataWeb: The Austrian building and construction materials market as linked data. In *Proceedings of the 9th International Conference on Semantic Systems—I-SEMANTICS '13*; ACM Press: New York, NY, USA, 2013; pp. 25–32.