Advanced Engineering Informatics 40 (2019) 14–28

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Full length article

Automated BIM data validation integrating open-standard schema with visual programming language

Pedram Ghannad[a], Yong-Cheol Lee[a,*], Johannes Dimyadi[b], Wawan Solihin[c]

[a] Department of Construction Management, Louisiana State University, Baton Rouge, LA, USA
[b] School of Computer Science, University of Auckland, New Zealand
[c] novaCITYNETS Pte. Ltd., Singapore

ARTICLE INFO

Keywords:
Building Information Modeling
BIM Data Checking
Visual programming language
LegalRuleML (LRML)

ABSTRACT

A building design must comply with a wide spectrum of requirements stipulated by building codes, normative standards, owner's specifications, industry's guidelines, and project requirements. The current rule-based compliance checking practice is a costly bottleneck in a building project, and thus, there is a demand for a design evaluation process that incorporates automated checking capabilities to address the inefficiency and the error-prone nature of the current manual checking practice. The inherent complexity of building design rules and impracticability of existing automated checking approaches are two key challenges that must be addressed to enable practical compliance checking automation. This research study proposes a new modularized framework that integrates the emerging open standard, LegalRuleML, with a Visual Programming Language. The framework allows a standardized method of defining design rules in a machine-readable and executable format. The proposed approach encompasses the entire compliance checking process from the interpretation of natural language-based requirements to machine-readable rules, rule categorization, rule parameterization, and the execution of the rules on the ISO-standard building information model. This modularized BIM-based design validation framework is expected to help automatically and iteratively evaluate the level of quality and defects of information conveyed in a given building model as an essential part of the early design process.

1. Introduction

There are numerous requirements, such as those stipulated by building codes, normative standards, owner's specifications, industry's guidelines, and project requirements that a building design must comply with before it is permitted to proceed to the construction phase. Most of these requirements can be represented as computable rules. Some are relatively straightforward, but some can be complex. Rule-checking is one of the most common approaches to assess a given design or project information for confirming compliance with a set of rules representing the requirements. Meticulous examination of a design against the requirements defined in design specifications is a key step that ensures all requirements are fulfilled. However, the conventional rule-checking practice is often a costly bottleneck that can negatively affect a project delivery process because it is laborious, time-consuming, and error-prone [16]. Automation of this process has been identified as having the potential of contributing significant time and cost savings to a project [7,18]. Because of its importance, several research studies regarding the automated compliance checking have been conducted since the 1960s and well before the advent of Building Information Modelling (BIM) [4]. The maturing of the BIM technology in recent decades has created diverse opportunities for sharing common building design information to automate rule-based compliance checking.

Rules can be represented in a natural language, formal notation, or a combination of both. One of the most fundamental factors for the successful development of the rule-checking process is an appropriate formalization of the multi-faceted aspects of norms, guidelines, and general legal knowledge into a machine-readable format. To represent normative information as computable rules for compliance checking purposes, various approaches have been suggested by researchers over the years [4]. Diverse closed proprietary systems exist [7], but some open standards such as LKIF (Legal Knowledge Interchange Standard) and LegalRuleML (LRML) have also emerged in recent years [5]. Utilizing open standards provides diverse benefits in promoting transparency, flexibility, and interoperability. Translation and validation tools developed to convert rules expressed in a natural language into an open standard computable form can be reused on any paper-based document.

* Corresponding author.
E-mail address: yclee@lsu.edu (Y.-C. Lee).

https://doi.org/10.1016/j.aei.2019.01.006
Received 11 September 2018; Received in revised form 20 January 2019; Accepted 29 January 2019
Available online 04 March 2019
1474-0346/ © 2019 Elsevier Ltd. All rights reserved.

In addition, deploying open standards can enable interoperability and facilitate data exchange.

In recent years, this form of representation has been adopted in various domains. Thus, the investigation of its application in the AEC industries is also helpful to tackle the challenges of complex requirements and the heterogeneous structure of a building project.

There are several approaches to automating compliance checking in the AEC domains, but they still have a considerable practical and technical gap. The current approaches tend to be inflexible and non-transparent, particularly with respect to rule definition and execution.

To enhance this situation, this study investigates a modularized rule-checking approach integrated with a visual programming language (VPL) using the LRML schema as the standard to represent building design rules.

The first part of the methodology is the collection and categorization of building compliant design requirements. They are then converted into rules in LRML to be executed using a selected VPL. For this study, the Marionette programming language in the Vectorworks software package is selected as the VPL tool. Vectorworks is one of the popular BIM authoring tools used in the AEC domains. The current scope of the study focuses on simple and fundamental design requirements needed during the early design phase. The case studies and the implementation details using the early design requirements are described in the design guidelines of the following project types: residential buildings, healthcare facilities, and educational complexes. They are validated to ensure the feasibility and the practicability of the proposed approach and assess the flexibility of a modularized rule-checking framework integrating VPL with LRML.

The module libraries developed on Marionette indicate that the proposed framework firmly offers a flexible and extensible rule-checking method that can handle multiple parameters and rule sets. This framework executes various checking features with a limited amount of checking nodes. The use of a node-based rule definition and analysis approach in VPL also elevates the transparency of the encoded rule system via the XML standard, which is readable and understandable for end users. This notable feature of the modularized rule-checking process offers users intuitive rule coordination opportunities that have not been achieved in previous rule-checking approaches

2. Background and literature review

2.1. Building design compliance checking

The compliance checking aspect is an integral part of the building design process to ensure that various requirements are fulfilled. More importantly, the compliance checking process plays an essential role throughout the entire life cycle of a building from inception to demolition. The traditional rule-checking practice in the AEC industries involves a manual process that is error-prone and inefficient. Over the last four decades, numerous studies have been conducted to figure out a practical solution for automating the compliance audit process [4]. According to the literature review, the authors have identified that the slow progress in the development of the automated rule-checking system has been caused by several factors such as the inherent complexity and the fragmented nature of the industry, a large number of stakeholders, resistance to change, and lack of motivation towards adopting new technologies [8,9].

The application of rule-based checking approaches has been investigated in diverse studies for evaluating a BIM design pertaining to compliance with computable rules. Eastman et al. [7] surveyed different types of rule-based compliance checking approaches, such as DesignCheck, SMARTCodes, ePlanCheck, and Solibri Model Checker (SMC). These methods provide rule-checking capabilities but entail a couple of major limitations. First, the rule-sets are generic in nature and predefined in a black-box environment that may or may not reflect a complete set of rules needed for any particular purpose. Second, a user is limited to customize the preset rules to satisfy their own purposes. SMC could be used for early design checking [13], but any permitted customization of predefined rule parameters requires a steep learning curve and the process is tedious and not user-friendly.

Lee et al. [14] also reviewed several rule-based platforms and applications such as dRofus, SMC, Revit reviewer add-on (SmartReview), and BIM Assure (developed by Invicara). One of the most common shortcomings of these systems is the inflexibility and the “black-box” approach. They do not allow users to define and execute their own rules as well as modify the pre-defined rulesets [17]. This limitation requires a system developer to consistently maintain the codes for every minor rule update or requirement addition, which make the process costly [6]. The lack of transparency is also one of the limitations of hard-coded systems resulting in domain experts not being able to accurately verify or validate the checking process.

One of the most challenging issues for building design compliance checking is the translation of human-readable rules into a computer-processable code. This obstacle has been examined by diverse researchers in recent years and most of studies have focused on the application of Natural Language Processing (NLP) and Artificial Intelligence (AI) techniques for automating the transformation process [34,28,35,32,12]. Although the progress in this research area is promising, well-defined codes have not yet been deployed for the validation of these AI-based approaches [26]. Furthermore, current normative requirements in a domain consist of complex interrelated legal documents that may contain contradictions and are often poorly structured. These documents are also subject to frequent changes and amendments, which make them challenging for automated translation into computable forms. The NLP and the AI techniques may provide a facility in keeping a computable representation updated from its source provision; however, the presence of an official data exchange standard seems to be the current gap that should be addressed [4].

Several studies aimed at developing a practical computer mark-up language to help the domain experts to encode legal text into computable rules. One of these initiatives was the SMARTCodes project undertaken by the International Code Council (ICC) back in the 2005. The document markup approach of manually encoding legal texts into computable rules has continued to be pursued by AEC3 through a revised technique known as RASE [10]. The RASE method of marking up legal texts and categorizing them into rule objects only provides a method of defining rules, but does not address a means of representing a set of rules that guarantee they reflect the latest amendment in the source provision.

The VPL method of encoding legal texts into rules [24,25] and [17] not only involves domain experts in the process, but also provides transparency, flexibility, and portability. The main concept of this approach is to leverage the visual and flow-based programming aspects to allow domain experts with limited knowledge of programming to work with the encoding process. Preidel and Borrmann developed the Visual Code Checking Language (VCCL) as a neutral approach. However, Lee et al. [17] worked with Marionette (the VPL embedded in Vectorworks, which is a BIM authoring tool) in order to take an advantage of the query capability within the BIM environment. The main focus of both studies is on the rule encoding and the rule validation process. However, the current research aims to tackle the entire code compliance checking process of a BIM Model with a consistent module schema-based rule translation and execution approach.

2.2. Formal representation and semantic interoperability using RuleML and LegalRuleML

For the domain of a building design, different forms of rulesets and norms such as international, national, local, and even project-specific rules may need to be complied with before, during and after a building is built. Rules are typically written in natural languages that should be interpreted before they can be analyzed by machines. Nature of a textual content is associated with various challenges such as exchanging specific

data between parties, searching for and extracting structured information within the text, and automatically doing further processes. Rule structures and clauses consist of patterns, but they entail detailed variations that make it hard to capture all required information. Experience with different rule applications and inductive reasoning is needed to extract such patterns, hidden assumptions, and dependencies with other rules [31]. Heterogeneous terminology or jargons and their representation structures are the primary challenges for exchanging legal data. In addition, since legal statements of each domain entail diverse intents, interests, and targeted realms, their representations are frequently abstract and unclear, leaving open-ended interpretations flexibly applicable in relevant cases and situations. Consequently, legislators, legal practitioners, and business managers have been impeded from comparing, contrasting, integrating, and reusing the contents of the texts. In the current web-enabled world, it becomes essential to provide machine-readable forms (generally in XML) of the contents of the text.

Formalization of the norms, guidelines, and general legal knowledge into a machine-readable format is a key enabling the successful development of a rule-checking system. To ameliorate this issue, one study has adopted the LRML to produce a rule interchange language for the legal domain [2]. Implementers can organize the contents of the legal texts in a machine-readable format by using the representational tools, which then feeds further processes of interchange, comparison, evaluating, and reasoning.

LRML [22] is built on RuleML (RML) [3], an open standard XML-based rule language, with an additional layer of features to address deontic operations (e.g., obligation, prohibitions, permissions, etc.) specific to legal norms. The development of this emerging open standard aims to enable the formalization of the logical content of norms, guidelines, and codes used in a variety of domains such as Engineering, Commerce, Law, etc. Metadata of rules and normative elements can also be captured by LRML. It also supports the legal isomorphism principle to maintain the connection between the rules and their source provisions represented by its complementary standard, LegalDocML [2].

Dimyadi et al. [5] evaluated LRML as a possible de-facto standard to express and share normative information in the AEC industries for automated processes. Their findings indicate that it can overcome the shortcomings of other standards since its features satisfy a set of preliminary criteria. These criteria are the readability for human and machines at the same time, the capability to represent logical and literal contents and maintain a tight-coupling with the source provision, the comprehensiveness that can represent any type of document, and using open standards to enhance the interoperability.

2.3. Graphical scripting or visual programming language in rule-checking

VPL can be described as a formal language with a graphical notation, which employs a modular system of signs and rules with visual elements instead of textual ones on the semantic and syntactic level [29]. VPL utilizes visual elements so that users can interpret it much faster and easier. Because the complexity of text-based programming languages makes the access for programming beginners difficult, visual (programming) languages such as Scratch [19] are used in the surrounding of teaching basic concepts of computer programming. In recent years, different building design software have established VPL as a part of their software products such as Grasshopper for Rhinoceros3D, Dynamo for Autodesk Revit, and Marionette for Vectorworks. Although these applications primarily concentrated on providing a useful tool for 3D parametric modeling, their features have been significantly extended by further functionalities [25]. VPL is flow-based and composed by a set of nodes. Each node generally represents a piece of a modularized code as well as a basic unit of programming. This graphical notation also has the capability to represent design rules in a machine- and human-readable language. The collection of nodes is similar to a network flowchart that provides the readability and the flexibility so that users can implement compliance checking efficiently [25]. Wülfing et al. [33] conducted a research study using VPL for querying information from a BIM model. They recognized the great potentials of VPL to retrieve information from a 5D-BIM which cannot be retrieved because of a lack of knowledge about the underlying data structures and the derivation of implicit and explicit information [30].

3. Methodology

To achieve the objectives of this research, the study proposes a framework for the BIM-based rule-checking process. The authors identified the following three core phases that can used for establishing VPL-based design compliance checking process: unstructured rule definition as natural language, formalized rule definition, and VPL-based rule representation. These phases can be implemented for all types of rules and all aspects of design rules such as semantics, ontologies, and logic. Each unit of the framework has been carefully designed in order to leverage the rule-checking procedure. Fig. 1 indicates the three phases for the design compliance checking process including unstructured rule definition as natural language (NL), formalized rule definition, and VPL-based rule representation.

The unique contribution of this framework is based on the identification of modules in each phase that provides a basis to map them to their related modules in a different phase of design compliance checking process. In other words, modules play a pivotal role in that they must be captured, tracked and transformed during the entire process. For example, a building element such as a wall is considered as a module that is a simple term in the natural language definition phase, a characterized and tagged word in the formalization phase, and finally a defined building element in a BIM model for the rule execution phase. In addition to the aforementioned advantages of the proposed framework, the modularization brings the benefits for the approach including the improvement of the traceability of rule-checking elements, which enhances the accuracy of design evaluation, reducing required time and effort by encouraging the reusability of modules [15].

Each phase has its own syntax rules that fall in the following two main categories, modules as tokens and structures. Modules are the basic elements of each phase such as words, numbers, signs, etc. The second type of the syntax rule pertains to the structure of a statement; it means the way the modules are arranged. The first phase is a natural language, in which most of the rules typically written. NL representation of rules is only human-readable and cannot be automatically processed by machines. Formalizing the rules and norms into a machine-readable format facilitates an analysis and processing of the rules. NLs are full of ambiguity that includes contextual clues and other abstract information. Formal languages are developed to provide nearly or completely unambiguous data representations. In addition, NLs employ considerable redundancy to make up for ambiguity and reduce misunderstandings, but formal languages are less redundant and more concise. The arrangement of modules (overlaps between modules in the NL phase) in Fig. 1 schematically shows the existing ambiguity and redundancy in NL representations of rules. In order to overcome the aforementioned challenges, formal representation of rules has been deployed as the second phase of the compliance checking process.

An appropriate approach for formalizing rules is a key factor for successful code compliance checking. A rule representation unit should be able to capture all elements of rules including semantics, ontologies, and logic regardless of the complexity of the rules. However, the inherent complexity of design rules in the current textual format does not allow to fully automate the rule translation with the current practices. Domain experts' knowledge is required to extract hidden assumptions embedded in design rules and norms. The XML-based standard has been proposed for this phase of our framework, formalizing logical content of norms, guidelines, and codes. Formalized rules can be automatically analyzed and processed to extract necessary information from a BIM design. Each rule consists of semantics related to each other by various logic to make the rule ontology. Identifying these parameters of rules

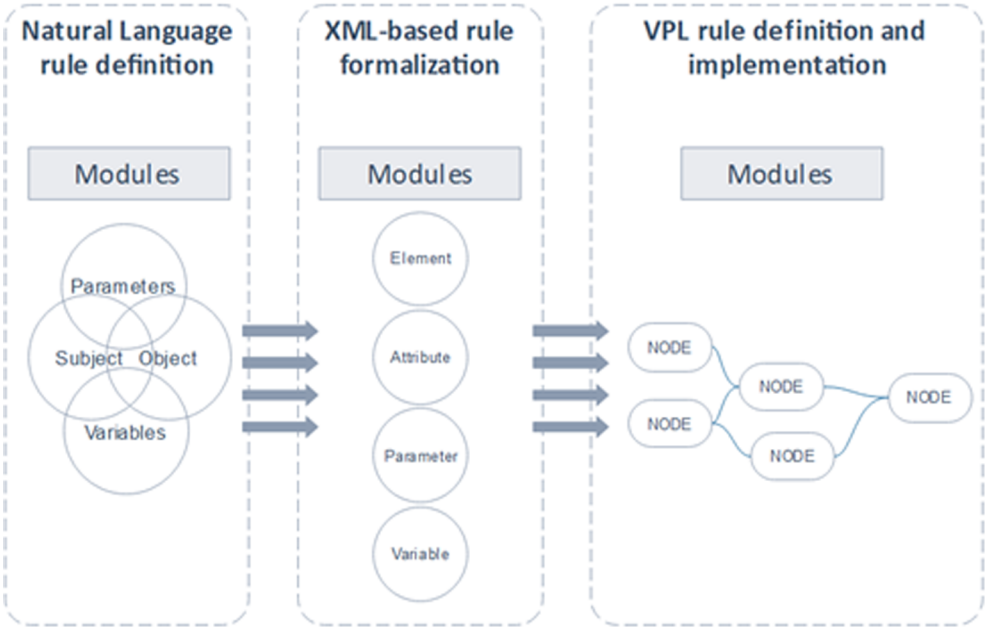

Fig. 1. Three phases of the proposed framework for design compliance checking.

facilitates the process of an information query required for the rule-checking process. It can also help domain experts flexibly and iteratively use rule-checking scenarios, ultimately reduce their time and efforts in design compliance checking.

The BIM-based rule-checking process needs to execute rule parameters against a BIM data model. VPL has been deployed for this transition phase, which addresses the data mapping process between the rule information and a BIM design. A precise transition process is one of the most pivotal factors for building a robust and automated rule-checking approach [20]. To accomplish this objective, this research team adopted VPL for developing a flexible and extensible platform for rule transition and implementation. According to Preidel et al. [27], VPLs have a remarkable potential for data analysis and processing tasks in the context of BIM. This feature of VPL in conjunction with its capability of data transition is a fundamental base for a design rule-checking procedure. VPL also allows users with minimum knowledge of programming to easily manipulate and control the process which is significantly effective to maintain the transparency of the approach.

Fig. 2 shows the following three layers of the proposed approach: the human-readable layer, the machine-computable layer, and the rule-checking layer.

The human-readable layer includes all documents that contain various types of codes, guidelines, and rules specified in a human natural language or tabular data requiring semantic and domain knowledge for interpretation. The first step in this research study is the collection of rules that should be checked in the early design phase. There are numerous codes and guidelines available at different levels such as national or state codes. To narrow down the scope of the rule collection phase, this study focuses on residential, educational, and healthcare facilities. The following three different documents are referred to examine early design rules: the International Residential Code (IRC) published by the International Code Council (ICC) (2015), the Guideline for Hospital and healthcare facilities published by the American Instate of Architects, and the State requirements for educational facilities published by Florida Department of Education. The analysis of the collected rules for the early design phase indicates that these rules are mostly about the spatial and geometric controls and the relationships between different building objects. The examples of these types of rules are evaluation about circulation, a room size, space sorting, a space ratio, a geometric property, and object existence.

The second layer shows formalized rule sets with the XML format, which is both human- and machine-readable. Transferring data between these two layers needs a considerable effort. Because of the complexity of rules, domain experts need to interpret and identify the hidden logic of the rules. In the next step, this research team deployed a schema to formalize the rules into a machine-readable format. The schema has been extracted from LRML and several simplifications have been conducted to improve its capability and practicability in accordance with the nature of building design rules. The schema is able to represent particularities of the rule in a building design with a rich and meaningful markup language. The most important features of rules that can be addressed by the schema are defeasibility of rules and defeasible logic, deontic operators (e.g., obligations, permissions, prohibitions, rights) and classification of rules (i.e., constitutive, prescriptive).

The third layer is the execution of the rule-checking process. This research study employed VPL on the BIM authoring platform, which allows users to intuitively generate and manipulate BIM objects in order to expose and use embedded building information. Using VPL also facilitates data transferring so that formalized rules in the XML format can be imported into VPL and used for mapping the rule statement content and rule-checking modules. VPL has the capability for parsing and extracting the required information from an XML instance file. Access to embedded building information and rule content simultaneously can make a robust platform to execute the rule-checking process against a BIM model.

In order to validate the proposed framework, the authors conducted several case studies by using Marionette, which is the VPL feature of Vectorworks. The integrated framework addresses the mapping of rule types, parameters, and reporting methods into each code and the execution of semi-automated rule-checking.

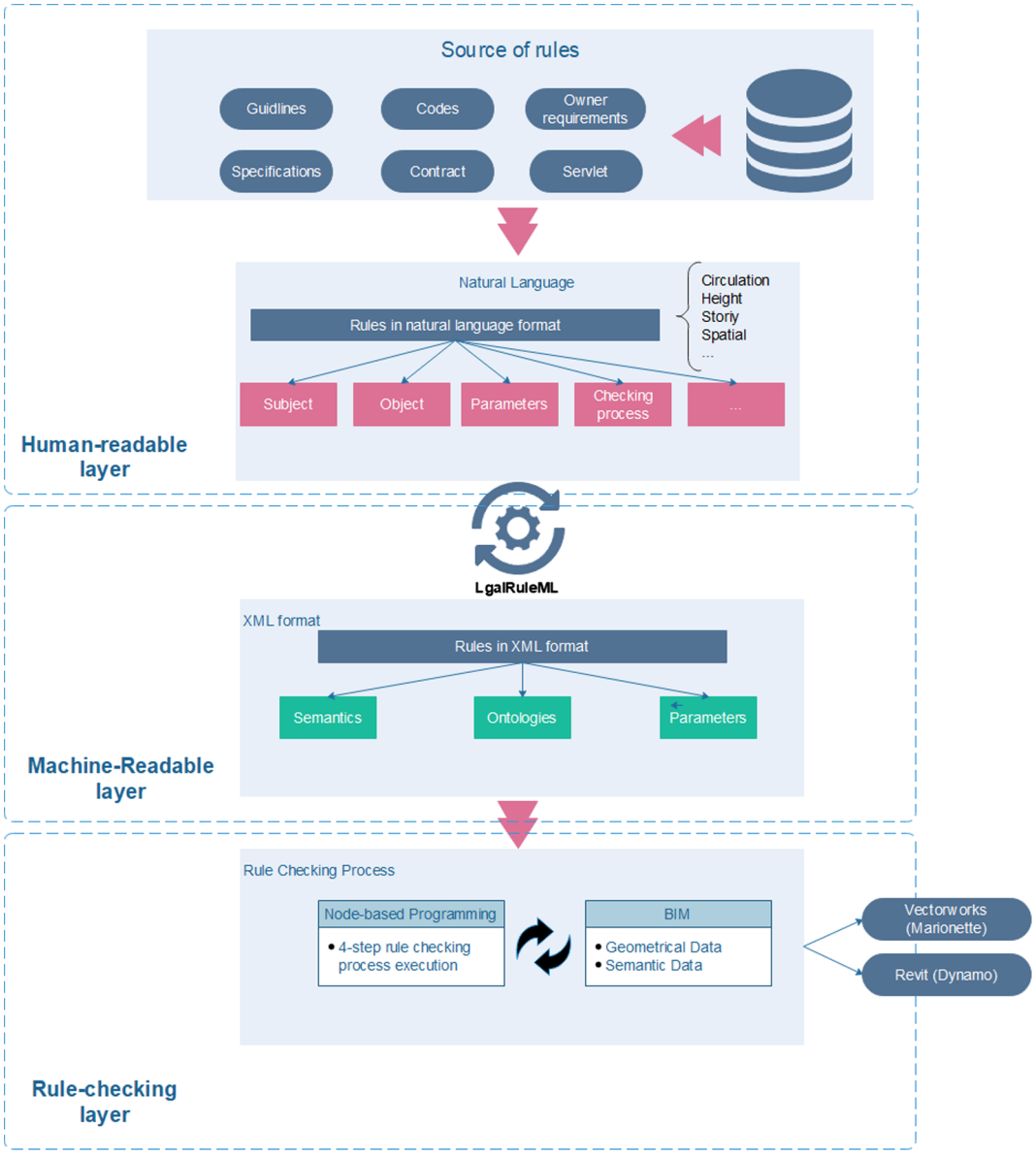

Fig. 2. Proposed LRML and Visual Programming-based Rule-checking framework.

4. LRML and VPL-based rule-checking framework

4.1. Human-readable layer and Machine-computable layer

The rule definition is the first major part of this rule-checking process. Rules are typically written in natural languages that need extensive domain knowledge and reasoning skills in order to interpret them into a machine-computable form. Although there are several ways to conduct the interpretation, most rule-checking studies focus merely on the language representation of rule syntax and grammar. In practice, expert knowledge is often required to interpret meanings or semantics of the rules such as intent, hidden assumptions, general knowledge of the subjects, and dependencies with other rules.

4.2. Rule classification and parametrization

Knowledge of domain experts is a key element to capture a hidden layer of rules and their interpretation. One of the major issues in automated rule-checking is addressing this interpretation process, which is currently carried out manually. The rule interpretation is a fundamental and challenging step in the design compliance checking. Solihin and Eastman [31] figured out the criteria of a rule classification defined by

Table 1
Rule parameters.

Rule Parameters	Description
Target Building Elements	Target object and subject of the rule or other building objects that are mentioned in the rule statement
Attributes	Characteristics of building objects that are used in the rule statement
Geometry	Geometry related attributes
Relationship	The relationship between two or more objects like inclusion, connection, and existence.
Logical Expressions	Expressions that indicate the logical relationship between two or more elements, statements, or variables, such as greater than, less than, And, Or, etc.

Table 2
Rule parametrization.

Rule parameters		Recommended Nodes and Modules
Target	Object	selection
	Space	Filter
Attributes	Position	getPosition
	Usage	getFunction
	Number	countNumber
	Material	getMaterial
	Name	getName
Geometry	Height	getHeight
	Length	getLength
	Width	getWidth
	Area	getArea
	Direction	getDirection
Relation	Inclusion	isIncludedIn
	Connection	isConnectedTo
	Existence	isExisting
Circulation	Accessibility	isAccessible
	Circulation Distance	getCirculationDist
Logic	Equal to	EqualTo
	Greater than	GreaterThan
	Less than	LessThan
	Negation	Boolean function
	And	linear network
	OR, XOR	parallel network

the following four general classes: (1) that require a small number of explicit data; (2) that require a simple derived attribute; (3) that require extended data structure; and (4) that require proof of solution.

According to the LRML principles, rules can be classified into the following two classes: prescriptive rules and constitutive rules. The constitutive rules provide the definitions of the terms and concepts used in a jurisdiction. On the other hand, the prescriptive rules dictate obligations, prohibitions, and permissions of a legal system.

A rule set consists of relevant criteria and required operations that are used to assess different aspects of a data model. In order to consistently organize and execute diverse rule sets, with a minimum number of nodes, this system classifies rule parameters based on their characteristics and attributes. The rule parameters of compliance checking needed in the early design phase are listed in Table 1.

Rule parametrization is an essential part of the rule-checking process because it makes a robust basis for an analysis of rules and rule-mapping processes to retrieve required information embedded in a BIM model. Accurate rule parametrization leads to a consistent rule mapping that decreases the effort of retrieving information and increases the accuracy of validating BIM data in the rule implementation phase at the same time. Table 2 contains the main categories of rule parameters. The second column shows possible types of rule parameters, but the types are not limited to the second column. The third column contains the recommended nodes and modules to address the related parameter, which can facilitate the rule mapping step.

4.3. Rules and LegalRuleML

The official schema of LRML published by OASIS [21] covers diverse aspects of normative representations beyond those needed for the early design phase. Therefore, only a subset of the schema is relevant to the work conducted in this study. The LRML standard can adequately represent particularities of rules used in the AEC domains [5]. The critical features that can be addressed by the schema include the defeasibility of rules and defeasible logic, deontic operators (e.g., obligations, permissions, prohibitions, rights) and classification of rules (i.e., constitutive, prescriptive). In addition, LRML supports isomorphism between rules and the natural language provisions that they represent. This feature provides a solid mechanism that verifies if a rule representation is valid in accordance with the latest version of the source provision.

According to the identified rule classification, most of the rules needed in the early design phase are modeled as: *IF* A_1, A_2,… A_n *THEN* C, with A_1, A_2,… A_n being pre-conditions and C is the effect of the rule. Consequently, each rule can be parsed to *IF* and *THEN* clauses. The LRML base schema can accommodate these two main parts of the rules and relevant elements as well as the attributes that can fully describe the contents of the rules. Table 3 shows the general scope of this rule representation in LRML.

The concept of defeasibility is incorporated into LRML to provide a means of resolving contradictions that may exist in a set of rules. LRML uses deontic operators to capture such notions. The above concepts have been extracted from LRML. However, the schema was modified in order to fully tackle the challenges of rule formalization. Based on the general logical format of design rules “IfClause” and “ThenClause” elements have been added to the schema as shown in Fig. 3.

Each of these elements contains a group element as the “Atom” element as shown in Fig. 4, which includes all parameters of a building rule identified in the rule parametrization section. The minimum occurrence of each parameter is zero and the maximum occurrence is unbounded, which means that a rule does not have to possess all parameters at once, and it can have an unbounded number of one parameter at once.

4.4. Rule mapping in LegalRuleML

In order to integrate LRML with the BIM-based rule-checking process, the authors mapped compliant design rules required for the early design phase into the rule classes in LRML. This rule mapping process offers not only a robust framework for domain professionals to design consistent and coherent rule-checking networks in VPL, but also a better understanding of the compliance checking process. Furthermore, this process is a promising step to move toward automation of the rule-checking process. Fig. 5 illustrates the schematic process of rule mapping.

Table 4 shows the example rule of a formalized normative provision: “Habitable rooms shall have a floor area of not less than 70 square feet (6.5 m^2).” Two main clauses are separated, and different rule parameters are underlined. The table also shows the mapping process for this rule statement based on defined rule parameters. For each parameter, the related module has been proposed to build the network for the rule-checking process. The mapped modules constitute the main structure of the network for the implementation of the compliance checking process.

Table 3
Organization of the general scope of rule representation in LRML.

	Defeasibility	Rule Class	Rule Deontic logic	Rule Components	Types of elements (sub-elements)
Rule statement	Defeasible Defeater Strict	Prescriptive Constitutive	Permission Obligation Prohibition	IF clause THEN clause	Building object Building space Attributes (Direction, Material, Position, Usage, Name) Geometry (Height, Length, Width, Depth, Area, etc.) Relationship (Inclusion, Connection, Existence, etc.) Variable

```
<xs:element name="IfClause" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:group ref="IfClauseStructure" />
  </xs:complexType>
</xs:element>
<xs:element name="ThenClause" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:group minOccurs="0" maxOccurs="unbounded" ref="ThenClauseStructure" />
  </xs:complexType>
</xs:element>
```

Fig. 3. IfClause and ThenClause elements.

```
<xs:group name="Atom">
  <xs:sequence>
    <xs:element name="TargetObject" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="TargetSpace" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="Attributes" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element name="Position" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="Usage" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="Material" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="Direction" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="Name" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="Att" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="Geometry" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="Relationship" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="LogicalExpression" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="Var" type="xs:anyType" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="VarUnit" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:group>
```

Fig. 4. Atom elements including all parameters of a building rule.

4.5. The rule-checking layer

In this research study, the following four steps required for rule definition and implementation have been identified as shown in Fig. 6: (1) object selection, (2) rule definition, (3) rule implementation, and (4) validation report. Marionette, one of the VPL platforms, has been used to implement these four steps because of the following two primary reasons. First, no extensive programming knowledge is needed for executing rule-checking processes with VPL. Designing and modifying a compliance checking scenario is easily applicable by domain experts.

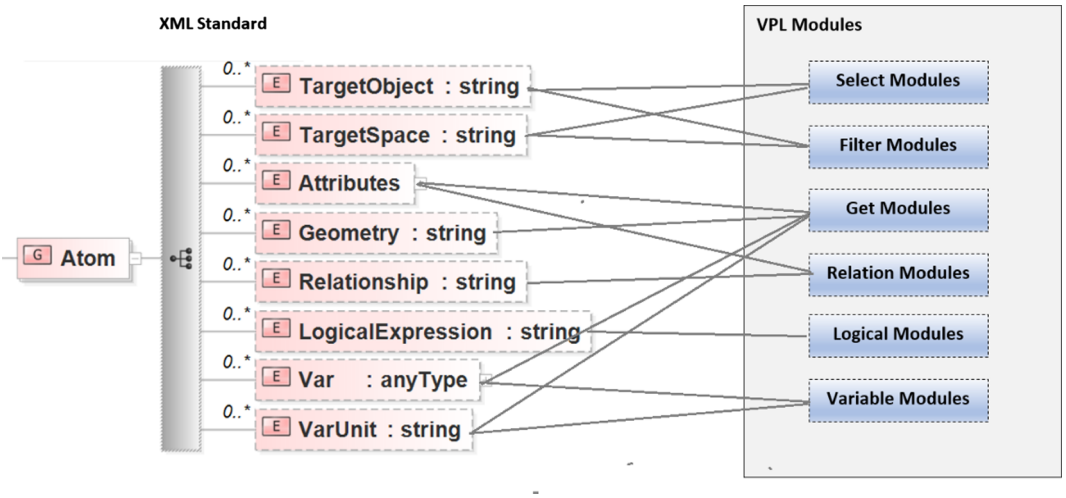

Fig. 5. The schematic rule mapping process.

Table 4
Rule mapping process.

XML instance	Module Type	Mapped Module
<PrescriptiveStatement RefID="IRCsecR304.1MinimumArea">		
<Rule>		
<IfClause>		
<Atom>		
<TargetObject>Space</TargetObject>	Filter Module and Select Module	
<Attributes>		
<Att>Function</Att>	Get Modules	
</Attributes>		
<LogicalExpression>EqualTo</LogicalExpression>	Logical Modules	
<Var>Habitable</Var>	Variable Modules	
</Atom>		
</IfClause>		
<ThenClause>		
<Obligation>		
<Atom>		
<Geometry>Area</Geometry>	Get Modules	
<LogicalExpression>GreaterThan</LogicalExpression>	Logical modules	
<Var>6.5</Var>	Variable Modules	
<VarUnit>SquareMeter</VarUnit>		
</Atom>		
</Obligation>		
</ThenClause>		
</Rule>		
</PrescriptiveStatement>		

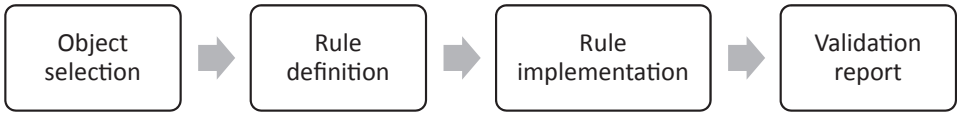

Fig. 6. Four steps of the rule-checking process in VPL.

Second, Marionette is embedded in the BIM authoring tool, Vectorworks, which can remarkably facilitate retrieving and querying BIM data required for rule-checking.

The first task of rule-checking is to select building elements evaluated by a rule that are not always a simple object. For example, in case of the following rule, "The ratio of height to thickness for all the walls shall not be smaller than 15," executing such a rule needs a module to

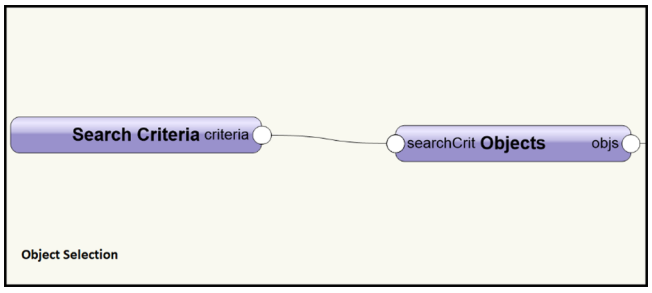

Fig. 7. Object selection step.

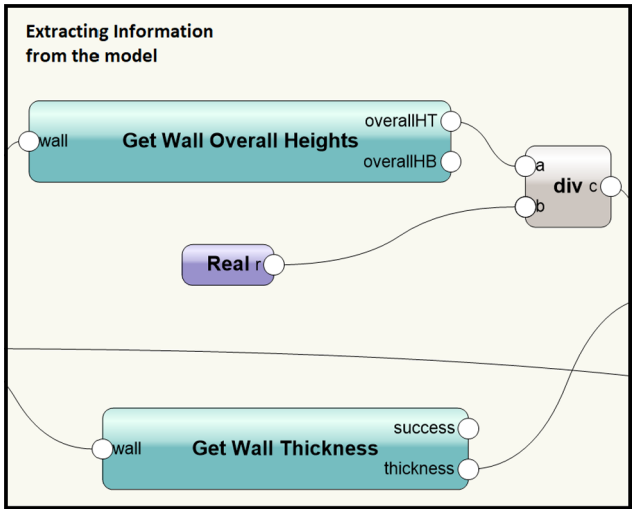

Fig. 8. Rule definition step.

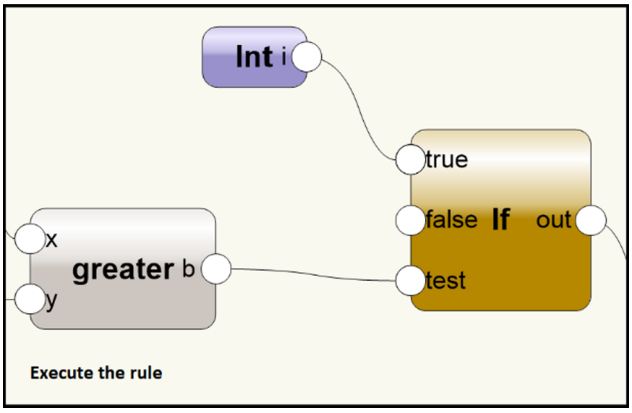

Fig. 9. Rule implementation step.

example, a logical node is used to check the required logical relationship "greater than" shown in Fig. 9.

After conducting the rule execution process, this framework produces a validation report clearly showing checking results. The validation report can be either in a visual or a text format. In order to create a report, this framework extracts the information of objects including tags, names, location, and other useful checking data. Fig. 10 indicates the validation report step in this case study. The "Get" module is utilized to make a visual report regarding the evaluated walls, which are not in compliance with the rule.

Fig. 11 indicates the entire network of nodes and their classifications in accordance with the proposed four-step rule-checking procedure (selection, definition, execution, and reporting) that finds out the wall objects that does not fulfill a defined ratio of height to thickness, 15, as shown in Fig. 12. The four steps provide a basis to develop and categorize VPL nodes and modules required for building and executing

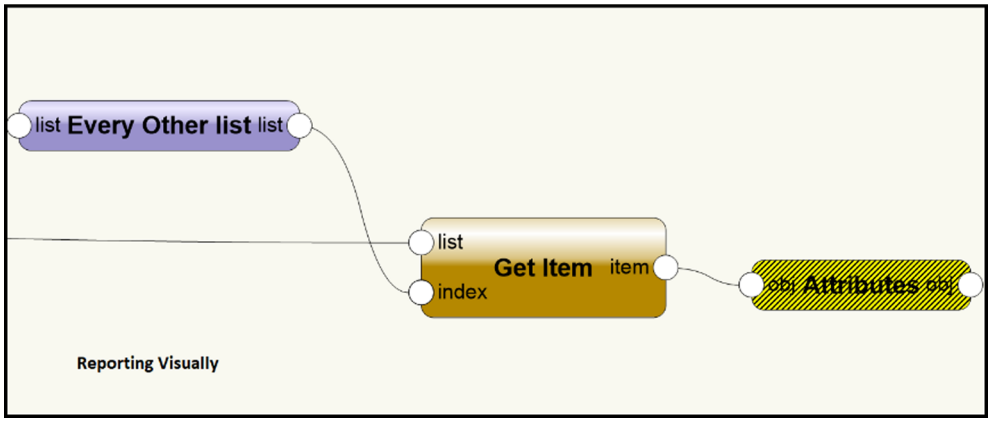

Fig. 10. Validation report step.

select the right elements (walls) in a BIM model. The object selection can be done by the "Filter" module and the "Select" module to collect all the walls of the BIM model as shown in Fig. 7.

The next step is to query required information in a BIM model by defining an implementable rule. Related information to rule parameters can be retrieved by the "Get" module and the ratio of the height of the wall to its thickness is calculated in this step as shown in Fig. 8.

Retrieved data and calculated information are deployed as inputs of the rule implementation step to execute the rule. Logical and relation modules provide primary functions to implement the rules. In this a rule-checking network. The modules are predefined nested nodes that can be utilized in each step of compliance checking to avoid complex and ambiguous networks. Modules are iteratively referred and implemented in several steps. The modules also have the capability to allow users to maintain the flexibility of the approach. Predefined modules are logical nodes, mathematical nodes, geometric nodes, and building model related nodes.

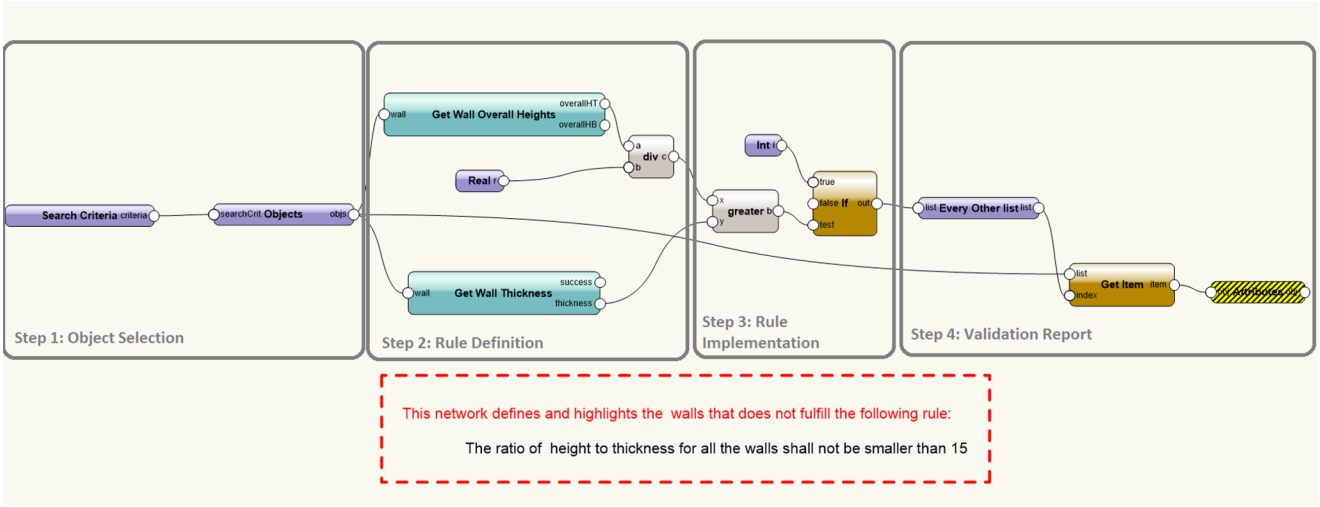

Fig. 11. The entire network of VPL-based rule execution.

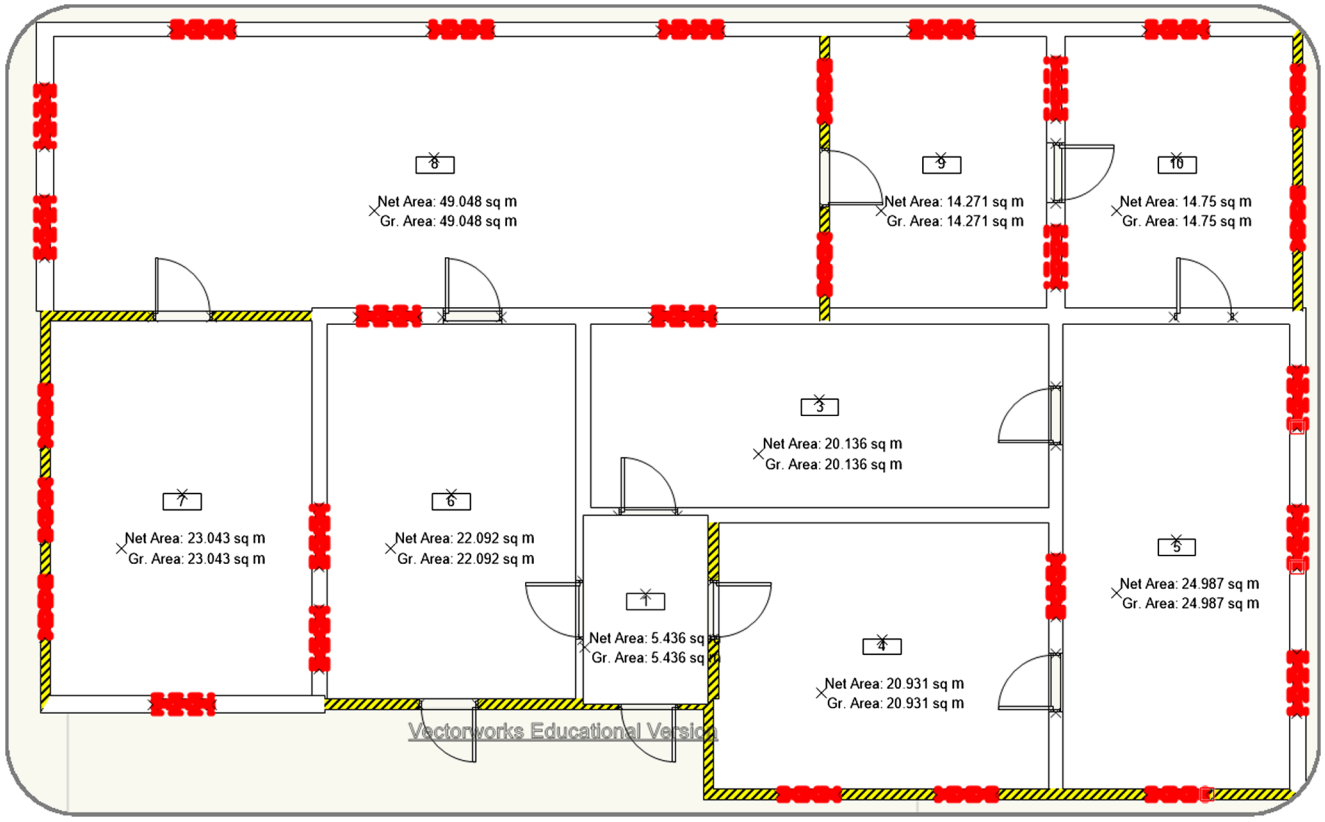

Fig. 12. Example of rule execution report according to four steps.

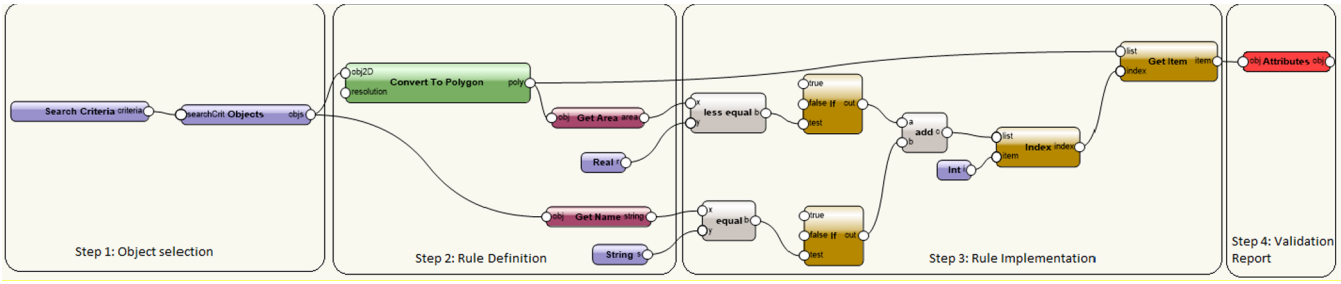

Fig. 13. Nodes network for checking minimum area rule.

5. Case study

The proposed semi-automated rule-checking approach is capable of executing the entire rule-checking process. To investigate the feasibility and the applicability of this approach, the authors conducted the case study using the following requirement of the early design phase specified in section R304.1 (Minimum room areas) of the International Residential Code [11]:

> "Habitable rooms shall have a floor area of not less than 70 square feet (6.5 m^2). Exception: Kitchens."

One single-story building model with multiple-rooms was designed as a test model for this case study. The above normative textual provision is translated into the XML format based on the DesignRuleML schema. The logical content of the above provision can be formalized into the following two rules, *"IRCsecR304.1Statements"* and *"IRCsecR304.1MinimumAreaException"* as Table 4.

The above rule statement has the one obligatory condition and the one permission condition: for example, if the function of a space is habitable, the area must be greater than 6.5 square meters. This rule can be defeated if the function of the habitable area is "kitchen." Table 4 also indicates the rule mapping between the XML format and VPL nodes, which contains the rule classification and parametrization. The output of this step involves the required modules for retrieving essential information from the data model and implementing the given rule-checking process.

Fig. 13 shows the node network designed for executing this rule. This network carries out the geometric property checking process in the following four steps: object selection, rule definition, rule implementation, and validation reporting. The output of this network is a visual report that highlights the spaces which do not fulfill the requirement as shown in Fig. 14.

In order to examine the capability of the proposed approach to deal with more complicated design rules, another case study was designed to execute the rule statement related to a circulation path in a healthcare building. This rule is specified in section 7.3. A1. Of the "Guidelines for Design and Construction of Hospital and Health Care Facilities" published by The American Institute of Architects Academy of Architecture for Health [1]. The rule states "The location of critical care unit shall offer convenient access from the emergency, respiratory therapy,

Fig. 14. Visual report for checking minimum area rule.

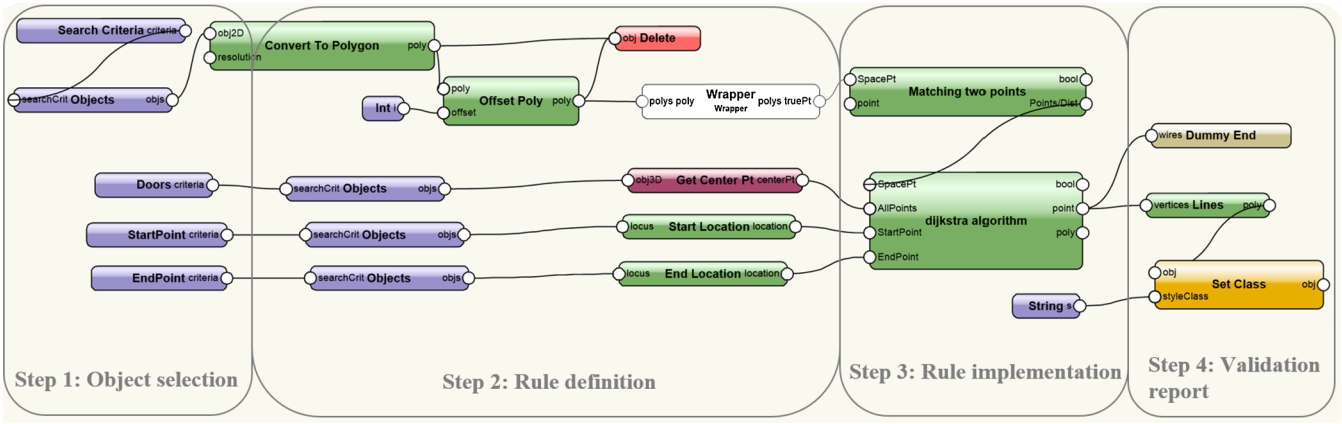

Fig. 15. Nodes for finding the shortest path between two selected points.

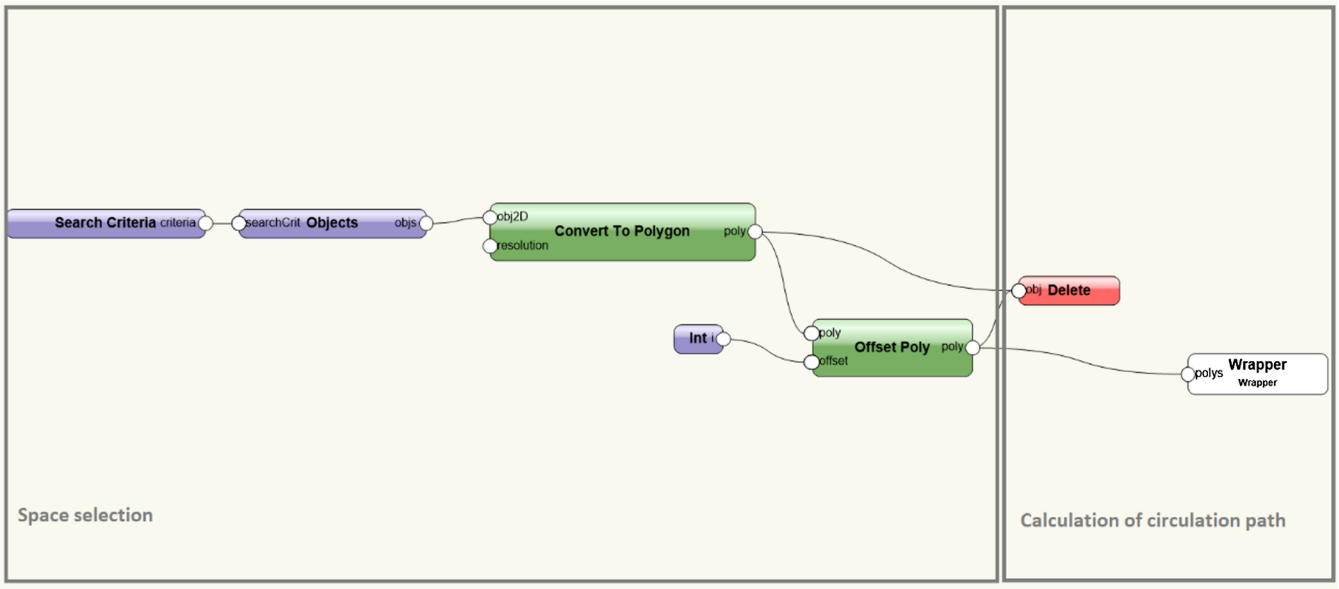

Fig. 16. Nodes for selecting objects and calculating circulation paths.

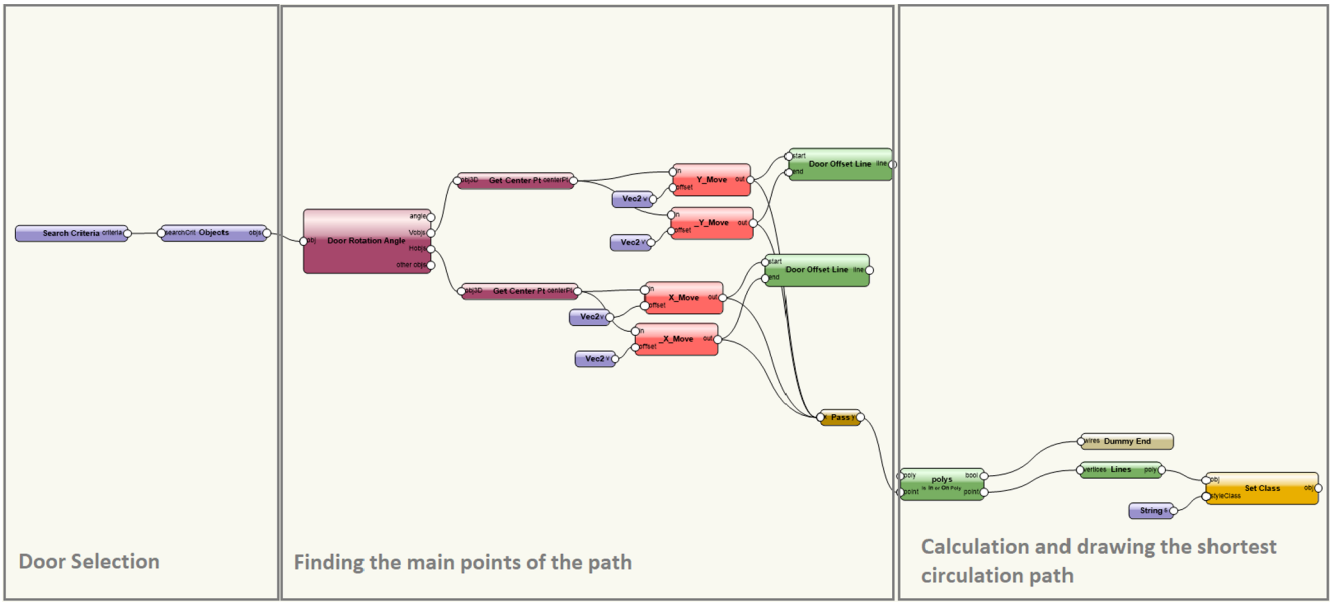

Fig. 17. Nodes for calculating the shortest path.

laboratory, radiology, surgery, and other essential departments and services as defined by the functional program. It shall be located so that the medical emergency resuscitation teams may be able to respond promptly to emergency calls within the minimum travel time. The location shall be arranged to eliminate the need for through traffic."

Despite the previous case study, there is no variable in this rule statement that shows a proper circulation distance for "convenient access" and "minimum travel time". Thus, in addition to the calculation of the circulation distance, the experience and the knowledge of domain experts must be involved to evaluate a design regarding the compliance with this rule statement. However, the calculation of circulation paths between different spaces is a time-consuming task. The proposed approach is able to facilitate this rule-checking process that needs explicit data retrieving and calculations.

The above rule contains various objects, subjects, and parameters and involves different spaces in a hospital classified by their functions such as "critical care unit", "laboratory", "radiology", and "surgery." Similar to the first case study, the "Filter" and the "Select" modules are

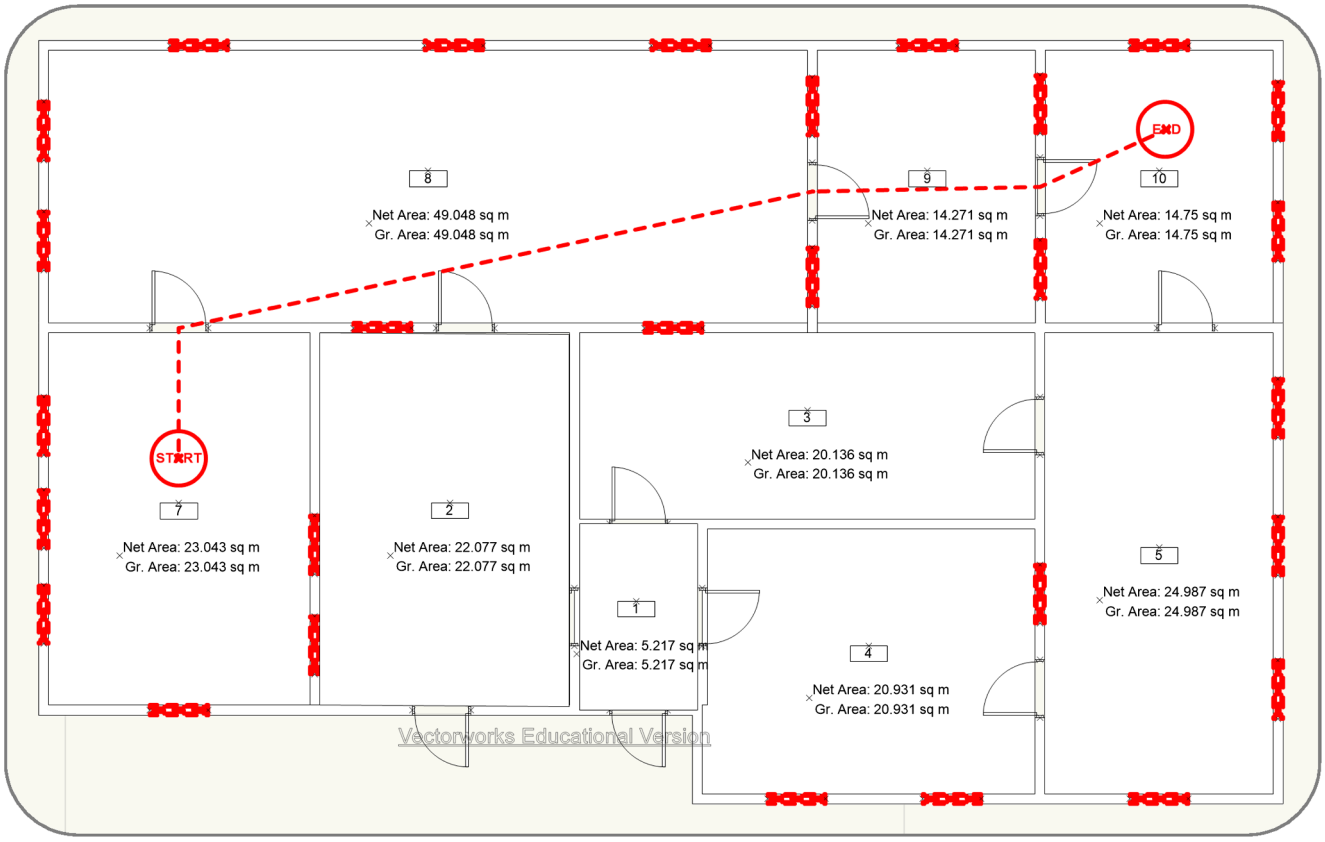

Fig. 18. The shortest path between two points.

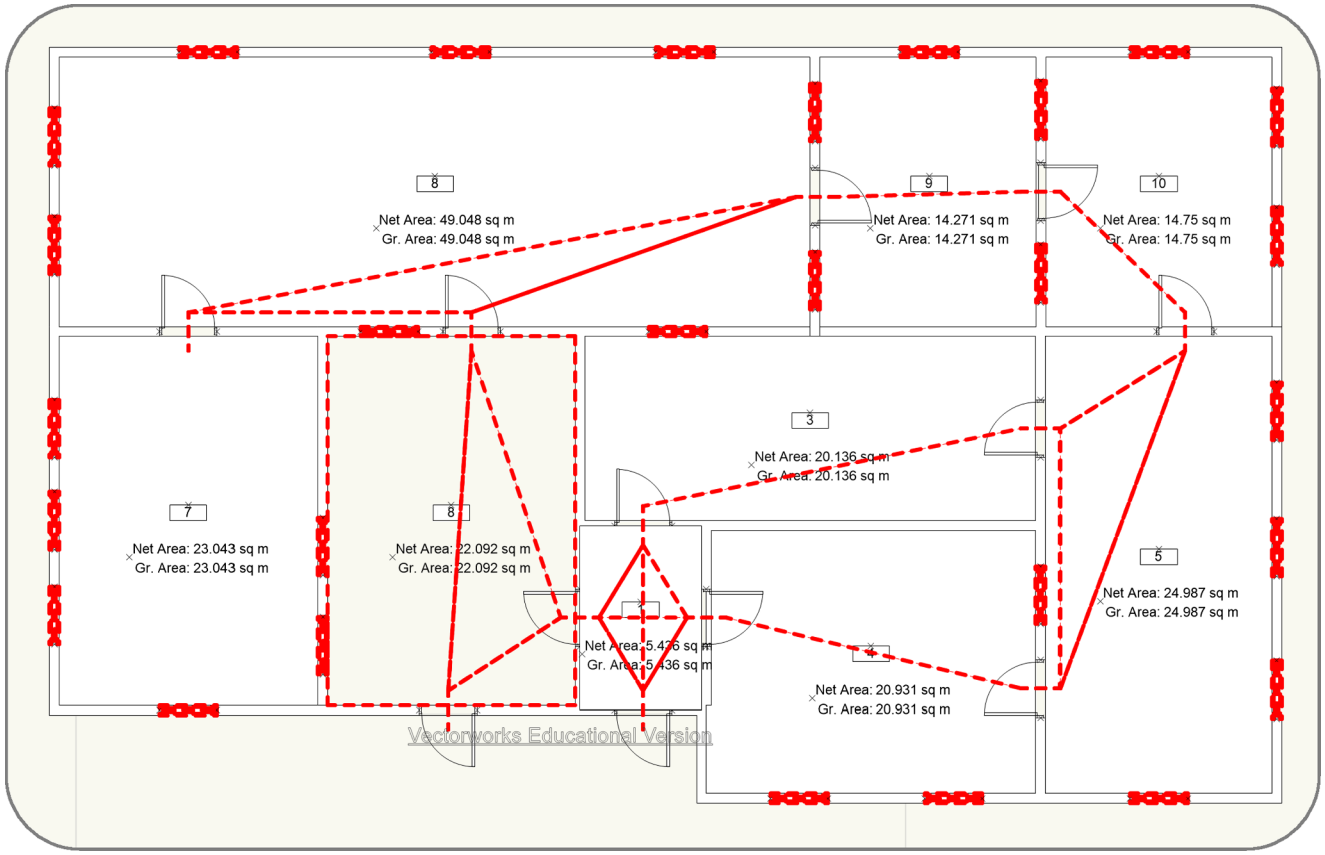

Fig. 19. Show all circulation paths.

used to query information from a BIM model. In addition, a predefined module in VPL has been developed for iteratively executing rule sets for the purpose of circulation checking.

The developed module, the node network shown in Fig. 15, calculates the shortest path between the two selected locations. This network implements the selection of the spaces and the identification of the door objects. The identified objects are calculated according to the Dijkstra algorithm [23], which is able to solve the single-source shortest path problem in weighted graphs. As shown in Fig. 16, the first step is the selection of spaces. Then, this network finds the effective points for the calculation of a circulation path, which is executed in a nested node “wrapper.”

In this step, this system calculates the shortest path between the two points in accordance with the available doors and routes. As shown in Fig. 17, the calculation of the shortest path involves the following three steps: (1) selection of the door objects and extraction of their information, such as their alignments and positions; (2) determining all the main points (center points) of the doors that can be used in calculation of circulation paths; and (3) calculation of all circulation paths by selecting the shortest one and setting it on the plan. Fig. 18 shows the calculated shortest path between the two selected points. In addition, Fig. 19 represents all circulation paths available in the floor plan. These two visual representations of circulation paths allow designers or relevant project participants to easily and instantly examine the design several times pertaining to diverse rule statements with the minimum effort.

6. Conclusion

Because of the extensive number of normative requirements and highly complex structures of building design models, the current practice of rule-checking in the AEC industries is conducted manually, including a time-consuming, error-prone, and costly process. The rule-checking process consists of repetitive tasks that can be automatically and rapidly performed by machines. However, most of the previous attempts to automate these processes have the inclination to hard-code rules into rule-checking systems. Using the "black-box" approach leads to several shortcomings such as the lack of transparency, the need for extensive knowledge of programming, and inflexibility. As a method to ameliorate this demand, this paper proposed a new modularized rulechecking approach that integrates a formal procedure of rule translation from NL-based compliant design requirements to the LRML-based formalized format with VPL-based rule execution. Our research team has identified the following three units of the design compliance checking process: natural language rule definition, schema-based formalized rule definition, and VPL-based rule implementation. A precise systematic rule mapping process is also proposed to link rule units to their equivalent modules in the three phases, facilitating the automation of the rule-checking process.

With these new findings, this study has revealed the benefits of the modularized design compliance checking process. The formalization phase provides considerable benefits by supporting consistent rule definition and checking processes with other project participants or domains. The XML-based rule formalization schema adopted in this study is readable and processable by various platforms and applications. Integration of VPL conducted for the rule implementation phase also enhances the flexibility of code compliance checking processes in rule definition and implementation.

Codes and guidelines in the AEC industries contain diverse statements that should be checked according to a large number of objects and elements in design models. VPL provides an opportunity for users to iteratively employ a single node network for checking similar rules with various objects in several models. The organization of networks can be easily manipulated for defining and executing rules with the same structure but different parameters.

Automated code compliance checking is a half-a-century long problem in the AEC industries that affects all aspects of the entire building’s lifecycle. This study focuses on the rules needed in the early design phase and therefore addresses a limited scope of rules. A further comprehensive study covering different types of rules in all design phases

and throughout the building's life-cycle will be conducted with this proposed approach as a future study of the authors. Another limitation of the proposed approach is the manual data transfer process between LRML and VPL, which requires users to manually organize nodes on the VPL platform according to the rules defined in LRML. Deploying machine learning methods in conjunction with the proposed approach is a promising method to provide the fully automated rule mapping process.

Appendix 1

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Created with Liquid Studio 2018 (https://www.liquid-technologies.com)-->
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Statement">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PrescriptiveStatement" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Rule" minOccurs="1" maxOccurs="1">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="IfClause" minOccurs="1" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:group ref="IfClauseStructure" />
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="ThenClause" minOccurs="1" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:group minOccurs="0" maxOccurs="unbounded" ref="ThenClauseStructure" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="RefID" type="xs:ID" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="RefID" type="xs:ID" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:group name="IfClauseStructure">
    <xs:sequence>
      <xs:element name="Atom" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:group minOccurs="0" maxOccurs="unbounded" ref="Atom" />
        </xs:complexType>
      </xs:element>
      <xs:element name="LogicalOperator" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:group>
  <xs:group name="ThenClauseStructure">
    <xs:sequence>
      <xs:element name="Obligation" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="Atom" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:group minOccurs="0" maxOccurs="unbounded" ref="Atom" />
              </xs:complexType>
            </xs:element>
            <xs:element name="LogicalOperator" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Permission" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="Atom" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:group minOccurs="0" maxOccurs="unbounded" ref="Atom" />
              </xs:complexType>
            </xs:element>
            <xs:element name="LogicalOperator" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Prohibition" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="Atom" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:group minOccurs="0" maxOccurs="unbounded" ref="Atom" />
```

```
                </xs:complexType>
              </xs:element>
              <xs:element name="LogicalOperator" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="LogicalOperator" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:group>
    <xs:group name="Atom">
      <xs:sequence>
        <xs:element name="TargetObject" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="TargetSpace" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="Attributes" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="Position" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="Usage" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="Material" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="Direction" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="Name" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
              <xs:element name="Att" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Geometry" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="Relationship" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="LogicalExpression" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="Var" type="xs:anyType" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="VarUnit" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:group>
</xs:schema>
```

Appendix A. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.aei.2019.01.006.

References

[1] American Institute of Architects Committee. Guidelines for design and construction of hospital and healthcare facilities, 2001.

[2] T. Athan, H. Boley, G. Governatori, M. Palmirani, A. Paschke, A. Wyner, Oasis legalruleml, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, ACM, 2013, pp. 3–12.

[3] H. Boley, A. Paschke, O. Shafiq, RuleML 1.0: the overarching specification of web rules, Springer, Berlin, Heidelberg, 2010, pp. 162–178.

[4] J. Dimyadi, R. Amor, Automated building code compliance checking–where is it at, Proc. CIB WBC (2013) 6.

[5] J. Dimyadi, G. Governatori, R. Amor, Evaluating LegalDocML and LegalRuleML as a Standard for Sharing Normative Information in the AEC/FM Domain, in: The Proceedings of the Joint Conference on Computing in Construction (JC3). Heraklion, Crete, Greece, 2017.

[6] J. Dimyadi, W. Solihin, C. Eastman, R. Amor, Integrating the BIM rule language into compliant design audit processes, in: Proceedings of the 33rd CIB W78 International Conference, 2016, pp. 1–10.

[7] C. Eastman, J.M. Lee, Y.S. Jeong, J.K. Lee, Automatic rule-based checking of building designs, Autom. Constr. 18 (8) (2009) 1011–1033.

[8] T.E. El-Diraby, S. Kinawy, Effective Semantic Web-based Solutions for Civil Engineering, in: Proceedings of CIB W78 Conference, 2008, pp. 1–6.

[9] T. Froese, Z. Han, M. Alldritt, Study of information technology development for the Canadian construction industry, Can. J. Civ. Eng. 34 (7) (2007) 817–829.

[10] E. Hjelseth, N. Nisbet, Capturing normative constraints by use of the semantic markup RASE methodology, in: Proceedings of the 28th International Conference of CIB W78, 2011, pp. 26–28.

[11] International Code Council, International Residential Code. Web page: < https://codes.iccsafe.org/public/document/toc/553/ > , Last accessed in June 2018, 2015.

[12] H. Lee, J.K. Lee, S. Park, I. Kim, Translating building legislation into a computer executable format for evaluating building permit requirements, Autom. Constr. 71 (2016) 49–61.

[13] Y.C. Lee, C.M. Eastman, J.K. Lee, Automated Rule-Based Checking for the Validation of Accessibility and Visibility of a Building Information Model, in Computing in Civil Engineering 2015, 2015, pp. 572–579.

[14] Y.C. Lee, C.M. Eastman, J.K. Lee, Validations for ensuring the interoperability of data exchange of a building information model, Autom. Constr. 58 (2015) 176–195.

[15] Y.C. Lee, C.M. Eastman, W. Solihin, R. See, Modularized rule-based validation of a BIM model pertaining to model views, Autom. Constr. 63 (2016) 1–11.

[16] Y.C. Lee, C.M. Eastman, W. Solihin, Logic for ensuring the data exchange integrity of building information models, Autom. Constr. 85 (2018) 249–262.

[17] Y.C. Lee, P. Ghannad, N. Shang, C. Eastman, S. Barrett, Graphical Scripting Approach Integrated with Speech Recognition for BIM-Based Rule Checking, in: Construction Research Congress 2018, 2018, pp. 262–272.

[18] Y.C. Lee, W. Solihin, C.M. Eastman, The mechanism and challenges of validating a building information model regarding data exchange standards, Autom. Constr. 100 (2019) 118–128.

[19] J. Maloney, M. Resnick, N. Rusk, B. Silverman, E. Eastmond, The Scratch programming language and environment, ACM Trans. Comput. Educ. (TOCE) 10 (4) (2010) 16.

[20] N. Nisbet, J. Wix, D. Conover, 17 The future of virtual construction and regulation checking, Virtual Futures for Design, Construction and Procurement, 2008, 241.

[21] OASIS,. LegalRuleML-Core-v1.0. Webpage: < http://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/cs01/legalruleml-core-spec-v1.0-cs01.html > (Last accessed in May 2018) 2018.

[22] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley, A. Paschke, LegalRuleML: XML-based rules and norms. In Rule-Based Modeling and Computing on the Semantic Web, Springer, Berlin, Heidelberg, 2011, pp. 298–312.

[23] S. Pemmaraju, S. Skiena, Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®, Cambridge University Press, 2003.

[24] C. Preidel, A. Borrmann, Automated code compliance checking based on a visual language and building information modeling, ISARC, Oulu, 2015.

[25] C. Preidel, A. Borrmann, Towards code compliance checking on the basis of a visual programming language, J. Inform. Technol. Constr. (ITcon) 21 (25) (2016) 402–421.

[26] Cornelius Preidel, André Borrmann, “BIM-Based Code Compliance Checking.” Building Information Modeling, Springer, Cham, 2018, 367–381.

[27] C. Preidel, S. Daum, A. Borrmann, Data retrieval from building information models based on visual programming, Visual. Eng. 5 (1) (2017) 18.

[28] D.M. Salama, N.M. El-Gohary, Semantic text classification for supporting automated compliance checking in construction, J. Comput. Civil Eng. 30 (1) (2016) 04014106.

[29] S. Schiffer, Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten Vol. 245 Addison-Wesley, München, 1998.

[30] W. Solihin, J. Dimyadi, Y.C. Lee, C. Eastman, R. Amor, The critical role of the accessible data for BIM based automated rule checking system, in: Proc. Lean & Computing in Construction Congress (LC3), 2017.

[31] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, Autom. Constr. 53 (2015) 69–82.

[32] M. Uhm, G. Lee, Y. Park, S. Kim, J. Jung, J.K. Lee, Requirements for computational rule checking of requests for proposals (RFPs) for building designs in South Korea, Adv. Eng. Inf. 29 (3) (2015) 602–615.

[33] A. Wülfing, R. Windisch, R.J. Scherer, A visual BIM query language, in: Proc 10th European Conference on Product and Process Modelling (ECPPM), 2014, August, pp. 157–164.

[34] J. Zhang, N.M. El-Gohary, Automated information transformation for automated regulatory compliance checking in construction, J. Comput. Civil Eng. 29 (4) (2015) B4015001.

[35] J. Zhang, N.M. El-Gohary, Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking, J. Comput. Civil Eng. 30 (2) (2016) 04015014–04015014.