BIM-SUPPORTED VISUAL LANGUAGE TO DEFINE BUILDING DESIGN REGULATIONS

HAYAN KIM1, JIN-KOOK LEE2, JAEYOUNG SHIN3 and JUNGSIK CHOI4
1,2,3 Hanyang University, South Korea
1,3 {hayaan92|jjyoung311}@gmail.com 2 designit@hanyang.ac.kr
4 Kyung Hee University, South Korea
4 jungsikchoi@khu.ac.kr

Abstract. Growing number of Building Information Modeling (BIM) applications have supported the automated assessment of building design and its quality in the early phase of design. For increasing the accuracy and fineness of assessment, rule interpretations require logical base and standardization of analysis process. Therefore, some government-funded research projects have focused on this rule-making process separated from the rule-checking process. Specifically, KBimLogic is a logic rule-based mechanism designed for the building permit related rules in Korea Building Act sentences. As a computer-readable definition of a rule, KBimCode has been developed to be executed in actual rule-checking software. The limitation of such code is the visibility to the rule experts who are usually non- or novice programmers. This paper describes much intuitive way of defining and generating KBimCode through KBim Visual Language. User can easily query the building element and method through the immanent connection with KBimLogic database. By using the KBim Visual Language, various types of rules written in design guideline, international standardization, and national acts can be easily interpreted into computer-readable formats such as KBimCode in order to proceed with the automated rule-checking.

Keywords. BIM (Building Information Modeling); Visual Language; Korea Building Act; Automated Design Process; Rule-making.

1. Introduction

Applications of Building Information Modeling (BIM) have affected the process of building permit to be partially automated by running the 3D building model-based rule-checking systems with related regulations. For the valid rule-checking process, an appropriate rule-making process for the computer-readable rules should
be done in logical interpretation without flaws. The conventional rule-making process primarily depends on the specific software and its own file format, therefore main part of the process is based on the closure method using script language-dependent hard coding process. Consequently, the primary steps of translating regulation are accomplished by software manufacturers or programmers. That makes the high threshold for architects or related parties who have little knowledge about programming language to check building related regulations.

The previous studies for the automated rule-checking have more stresses on the execution algorithms compatible with the IFC (Industry Foundation Classes) structure, often using hard-coded rules. However, for dealing with a broad range of rules in AEC-FM (architecture, engineering, construction and facility management) domain, logical interpretation based on syntactic, lexical, and semantic aspects should be ahead of the rule-making process. Therefore, in the checking software, the translated rules have to be logically valid and reflect the user intention as well. Because of the specific software development environment, only software experts could generate and modify the rules with their own definition. As a result, for the computer executable building rule-making process necessarily needs both architects and programmers to prevent the error-prone translating process.

As the reasonable alternative, this paper introduces BIM-supported visual language-KBim Visual Language (KBVL)- for rule-making which can be handled by the people who are usually non- or novice programmers. The logic rule-based mechanism called KBimLogic and its output KBimCode (Lee et al. 2016) are the basis of a logical structure for the KBVL. Also, KBVL utilizes the meta-databases established as a part of KBimLogic project, including the exact vocabulary of building elements and methods derived from the Korea Building Act.

2. Visual Language for Building Rule-making

Visual language emerged as an object-oriented programming language for developing a delicate algorithm and modifying the result in a parametric way. This also expresses the information as graphical symbols to be easily understood and used by users (Costagliola et al. 2002). Each graphical symbols have both semantic and syntactic attributes and are composed according to the structure definition of language. By the graphical representation, visual language is easy to learn, intuitive to use, and unnecessary for professional knowledge about programming. Consequently, the visual language supports managing information with visual elements and generating parametric logical rules with a complex structure in an intuitive way. In this regard, the visual proceeding makes possible to overcome the limitation of script-language-based hard coding method to reflect complicate relation between building regulations.

Thanks to the high usability of visual language, composing or checking complex contents with intricate relation became approachable. Since the building related regulations are involved with complicated relations between each building elements and include complex provisions, the visual representation is effective to grasp the contents of regulations.

There have been related projects about developing visual language-based build-
BIM-SUPPORTED VISUAL LANGUAGE TO DEFINE BUILDING DESIGN REGULATIONS

VBQL (Visual Bim Query Language) is for information extraction with filter query especially for craftsmen (Wülfing et al. 2014). As the result of eWorkBau project in Germany, this language could extract specific information from the 5D BIM model. Also, as a similar language, QL4BIM (Query Language for BIM) was developed to simplify the process of 4D BIM analysis. It supports even non-expert about programming to compose node combination to define queries as an intuitive way (Daum et al. 2015). As an expansion of the scope of querying information, language for rule-making and checking such as VCCL (Visual Code Checking Language) was developed for representing the result of building code checking with visual elements (Preidela et al. 2015-2016).

Preceded studies have been mostly focused on querying and checking the simple geometric specification of the building elements. The KBVL differentiates its features by utilizing the database of KBimLogic based on the Korea Building Act. Every element is prescribed in the database by the definition in the Korea Building Act. The semantic querying of visual elements from database decreases the feasibility of logical redundancy or omission between each node among the rule-making process. By standardized definition, user defined rule or any different kind of regulations can be composed of refined contents.

3. Logical Background for KBim Visual Language

This section describes the logical background for KBVL. As shown in figure 1, KBimCode ruleset file deduced as the result of composing components from KBimLogic-meta database. KBVL which for fasten the composition process of KBimCode, visually defines the relation of components by the connections and nodes. Each visual symbol uses the logic structure that already defined as KBimLogic and the definition of components defined in the meta database.

![Figure 1. Overview of KBimLogic-translating Korea Building Act into KBimCode.](image)

3.1. GENERATE KBIMCODE BASED ON KBIMLOGIC

To decrease logical error among the user defined KBVL, it puts its basis on the KBimLogic, the logic rule-based mechanism for translating building permit related Korea Building Act into the form of computer-executable format called KBimCode. By the KBimLogic, the rule sentences are split into the irreducible unit to increase accuracy and precision of representing the logical relation between
regulations. As the result, about 2,200 articles of building permit related regulation in Korea Building Act are decomposed into about 15,300 sentences and collected in the database for querying by the relation of other sentences. From the sentences (1) building elements and (2) their properties, (3) logical methods from verb phrase of sentences, and (4) relations between each sentence are extracted and collected as a database called KBimLogic meta-database.

This logic-rule based mechanism and the meta-database are the basis for the structure of BIM-supportive KBimCode by the object mapping with IFC model. The KBimCodes are composed of others according to the relation -commission, and reference- described in the Korea Building Act. As the result of composing, the ruleset file that including complex regulation definition and their relation is generated as computer executable format.

3.2. KBIMLOGIC FOR KBVL

Different with KBimCode which is basically translating Korea Building Acts, KBVL covers various types of regulations that defined by the user. For expanding KBimLogic to the KBVL, KBim meta-database offers standardized library for compounding visual language combination. The objects and properties with specific action that logically predefined in the KBimLogic meta-database are used as a visual element of the KBVL. Since the database is composed of analyzed and gathered components from the Korea Building Act, utilizing database could minimize the fallacy of definition in each visual symbol and increase the reliability of KBVL composition.

KBVL is generated by representing the combination of visual elements by queried attributes from KBimLogic meta-database, and automatically descript KBimCode at the background. By the KBVL, the user easily generates the complicated KBimCode by simplified visual symbols. Automatically generated KBimCode can be exported as the ruleset file to be used in the other rule-checking software.

4. KBim Visual Language

KBVL supports users to understand and implement the building permit related rule-making process without expert knowledge about script-language-based coding system. By the arrangement of visual symbols that already defined and originated in the KBimLogic meta-database, the user can easily compose standardized KBimCode.

4.1. BASIC COMPONENTS OF THE KBIM VISUAL LANGUAGE

The basic components mean two main component - node and connection. The nodes mainly include and symbolize specific information or function to define the action of the data. The connections of nodes automatically generate KBimCode in compliance with its grammar.
4.1.1. Node component

Node is a basic element of the visual language in order to reveal semantic information as well as to exact elements of the building model. As shown in figure 2, the general KBVL node has two kinds of port to import or export the data, title, and content of the node. Based on the distinctive function and feature of each node, this part describes four types of nodes. Each line of content reveals or requires the specific types of information. If the appropriate type of input is entered, color of the input port become black as verifying the connection between nodes, but white color means that nothing is entered and generated through the port.

![Figure 2. The visual attributes of the general node.](image)

- **Method node:** Method node plays a role of defining an exact action on the inputted object by connection with KBimLogic meta-database. By the linkage with the database, each method has a different action, type and the number of parameters. Based on its classification according to the attributes, basic rule module requires several object nodes to be connected to the input port of the method node, and additional return value node. Table 1 shows appearance features and using examples according to the different return value of the method. Inputted object is treated with the action of the method and the result is described on the method node or additional node. The difference between the method with Boolean return value and the others is whether it requires the return value node. The method with Boolean return value reveals itself the negation as dotted edge line. The other nodes such as the method of collection or numeric value need an additional node which appoints the target value.

- **Object and property node:** As input values for the method node, object and property have to be entered through the input port according to the different types of parameters of the method. Since this node has only the function of querying specific building elements, there is only an output port to be connected with the method node. The another characteristic of the object and property node is that there are several properties which are subordinated with the object and represented as a dotted line in the node content’s line by line. Exact querying and mapping occur between the building elements based on the KBimLogic meta-database.

- **Return value node:** As mentioned in the method node section, there are several types of method nodes which are needed to return a value node. The return value node has two types of data type - string and numeric representation. To assign specific collection or numeric value, the return value node is connected to the output port of the method node. The result of method node requires having the value of return value node.
Logic node: Rule-making process has a logical structure based on the CS-KS relation between each rule module. Definition of precise link relation decreases reiteration and unnecessary phase which may cause complicated structure of visual language and delay of the rule-making process. The logic node defines the Condition-Result-Else relation between the rule modules through the output port of the node. The node includes user setting option of adding or deleting the number of input or output port. In the part of ‘Condition’ section, CS (Condition Statement) is entered as a requirement of checking. ‘Result’ section includes KS (Key Statement) if the result of conditional checking is true. If the result is false, the result statement is described in the ‘Else’ section.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Return Value</th>
<th>Using Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>method() Parameter1</td>
<td>String: Methods querying the name of collection.</td>
<td>getBuildingUsage(), getObjectName(), …</td>
</tr>
<tr>
<td>Example</td>
<td></td>
<td>MultiUnitHouse</td>
</tr>
<tr>
<td>method() Parameter1</td>
<td>Boolean: Methods distinguishing result between true and false.</td>
<td>isExist(), isOpenObjectProperty(), …</td>
</tr>
<tr>
<td>Example</td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>method() Parameter1</td>
<td>Numeric: Methods querying numeric value (number, count, height …).</td>
<td>getObjectCount(), getFloorNumber(), …</td>
</tr>
<tr>
<td>Example</td>
<td></td>
<td>getFloorNumber()</td>
</tr>
</tbody>
</table>

4.1.2. Connection component

The connections of the nodes basically represent the flow of information and complete the node combination of rule modules with complex reference. The connection has two types of representation- solid line and dotted line. The solid line represents the flow of data and connection to the other node through the input port. The other type, dotted line shows ‘OR’ relation between several nodes. Each independent node combination implies ‘AND’ relation.

4.2. BASIC RULE MODULE

To generate a basic rule module, at least one combination of the method node with proper object and the return value node are necessary. To evaluated specific and delicate element, CS-KS linkage has to be refined to remove ambiguity involved in contents. The process of fragmentizing the contents of rule as basic rule modules and composing them again as bigger rule module supports making detailed relational indexes which could have been missed. The structure of basic rule module is based on the ALU (Arithmetic Logic Unit) for logical execution. Each basic rule module is composed of a single method node, its object, and property node, return value node according to the type of method, and the connection between each node.
4.3. NESTING COMPONENT

The nesting relation starts with the basic rule module which is the fundamental level of checking implementation. Each rule module is bound in a box-shaped container as another checking module. To evaluate complicated and delicate elements, several single basic rule modules are connected with other modules. Single node connection also can generate logical rule but through the nested component, repetitive usage of the same node is decreased and detailed checking indexes that could have been missed can be handled without omission. If contents in the checking process include multiple aspects with negation and logical connections, nesting relation can effectively offer intuitive visualization of rule module and connection relation. For example, “myWindow” component is nested with a basic rule module which means the window of which the panel operation type is a swinging window (figure 4).

5. Pilot Test

The pilot test is for verifying the possibility of translating various type of regulations by the KBVL. In this section, the part of Enforcement Decree of the Building Act 35-1 (EDBA 35-1) is represented with KBVL. The original text of EDBA 35-1 is:

- **Enforcement Decree of the Building Act 35-1 (EDBA 35-1):** (Installation of Fire Escape Stairs) Direct stairs installed on the fifth or upper floor or the second or lower underground floor under Article 49 (1) of the Act shall be installed as fire escape stairs or special escape stairs according to the standards prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport.

As in figure 5, for the compact node combination, the specific user-defined object should be predefined by using bound box component. Composed components are used in the overall node combination to maintain and reveal the specific condition of the user defined object. Figure 6 shows CS-KS classification of EDBA 35-1
contents to verify the logical relation between them. Through binding components, each CS and KS nested as single basic rule module and connected with logic node shown in figure 7.

![Figure 5. Definition of the specific component “myFloor”, and “myStair”.](image)

[CS3] Direct stairs installed on the fifth or upper floor or the second or lower underground floor under Article 49 (1) of the Act shall be [KS5] installed as fire escape stairs or special escape stairs

[KS4] According to the standards prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport: Provided, That the same shall not apply to cases where [CS1] main structural parts are made of a fireproof structure or noncombustible materials and [CS2] falls under any of the following subparagraphs.

![Figure 6. CS-KS classification of EDBA 35-1 contents.](image)

Figure 6. CS-KS classification of EDBA 35-1 contents.

[CS3] Direct stairs installed on the fifth or upper floor or the second or lower underground floor under Article 49 (1) of the Act shall be [KS5] installed as fire escape stairs or special escape stairs

[KS4] According to the standards prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport: Provided, That the same shall not apply to cases where [CS1] main structural parts are made of a fireproof structure or noncombustible materials and [CS2] falls under any of the following subparagraphs.

As the result of combining KBVL, the final logic node generates the KBim-Code to be extracted as the ruleset file for the execution of the other rule-checking software.
6. Conclusion

This paper describes a logical basis and structure element of BIM-supported visual language-KBVL to define the part of building design regulations. To verify the availability of the KBVL, the pilot test proceeded for representing actual rules into the KBVL. As the logical base, KBVL based on the KBimLogic, the logic rule-based mechanism to convert Korea Building Act into the form of computer executable format. As the structure elements, each node expresses the queried semantic attributes from the KBimLogic meta-database. Through the management of the database, KBVL composition also can be managed with standardized contents. The connection of each method node with the required object and return value node generates basic rule module for executing fundamental rule-checking. By composing and nesting several basic rule module, complicated rule can be represented with intuitive visual elements. The conventional rule-making process that based on scripting hard coding is now easily accessible with the composing of the visual element. This research also expanding its application range to not only for the Korea Building Act but also other kinds of regulations such as design guideline and international standard document to cover various type of building design related regulation.

Acknowledgements

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2015R1C1A1A01053497).
References


