KBimCode-based Applications for the Representation, Definition and Evaluation of Building Permit Rules

Seokyung Park[a] and Jin-Kook Lee[a]

[a]Dept of Interior Architecture Design, Hanyang University, Korea
E-mail: seokyung.park529@gmail.com, designitlab@hanyang.ac.kr

Abstract –

This paper aims to describe applications of KBimCode that enables representation, definition and evaluation of building permit rules for BIM-enabled code compliance checking. KBimCode is a standardized script language. It is developed to represent the Korean Building Act as explicit computable rules. The application of KBimCode can be summarized as: 1) Preparation of natural language rule, 2) representation of rule, 3) definition of rule, 4) execution of rule, and 5) execution result and report. This paper mainly focuses on describing the process and mechanism of each stage of KBimCode application. First, natural language rules are translated into KBimCode through logic rule-based mechanism. KBimCode are generated and managed by logic rule-based system called KBimLogic. Second, we introduce a design strategy and definition of KBimCode syntax. Application of KBimCode database is also handled. Third, execution of KBimCode is demonstrated with BIM assessment software and its result is reported. For the demonstration presented in this paper, Korean Building Act sentences were translated into KBimCode series and evaluated in a BIM assessment tool named KbimAssess-lite. KBimCode is an ongoing project and will contribute to establish automated design quality assessment system for building permit in Korea.

Keywords –

Automated code compliance checking; Automated Building Permit System; Korean Building Act; KBimCode; KBimLogic; BIM (Building Information Modeling)

1 Introduction

Assessment of building design is an important task in every stage of design process to improve the entire quality of design, meeting owner's requirement, or getting a building permit [1]. Various design requirements have been reviewed by building designers in manual. Recently, as building information modelling wields great influence upon architecture, engineering, construction and facility management (AEC-FM) industry, research on BIM-enabled automatic code compliance checking has been explored by domain experts. Some challenging projects and research reported that BIM-enabled code compliance checking benefits conventional assessment process by improving efficiency of the process, reducing errors, etc. [2, 3, 4, 5, 6]. Among various issues related to the BIM-enabled code compliance checking, this paper focuses on interpretation of design rule and its execution. Issues related to building model preparation and rule checking reports are handled with less importance. Design rules are written in natural language, therefore, they need to be translated into a computable format that enables automatic reasoning. This paper points out how design rules can be assessed in a BIM environment centered on KBimCode. KBimCode is a standardized script language that represents natural language rule into a computable format. It was developed as an outcome of ongoing government-funded research and development project for establishing an automated building permit system in Korea [7]. We describe an approach to representing, defining and evaluating building permit requirement in Korea Building Act based on application of KBimCode.

2 Background

2.1 Researches in Rule-based Checking of Building Design

There have been challenging attempts to automate code compliance checking process in the field of the AEC-FM industry. The tracks of research activities in this area date back to the early 1960s when Fenves formalized specifications of the American Institute of Steel Construction (AISC) using decision table for automatic reasoning [8]. Recent studies and projects showed the possibility of various approaches with the issue. The CORENET (COnstruction and Real Estate NETwork) project in Singapore translated the Building Code of Singapore by hardcoding it with a programming language [9]. The USA General Services Administration's courthouse project parameterized circulation rules according to a predefined parametric table [10]. Lee [11] developed a domain-specific language named Building Environment and Rule Analysis (BERA) to analyze and define building circulation and spatial programming rules. The DesignCheck project of Australia formalized the Building Code of Australia and encoded it according to the EDM rule schema based on the EXPRESS language [12]; Uhm et al. analyzed RFPs for large public buildings in Korea and formalized them using four morphemes (object, methods, level of strictness, and others). The formalized rules were translated into the semantic web rule language (SWRL) and translated into the Python script language [13]. Zhang et al. developed algorithms that automatically extract information from the International Building Code (IBC) and transform it into a formalized format using the NLP approach [14, 15].

The existing studies showed various approaches of rule-making in the design assessment. However, there still remain limitations to apply existing approaches to the translation of the Korean Building Act. Korean Building Act inheres complex relationships between sentences, therefore, new approach reflecting such feature is needed. In addition, because it is national binding force and errors in the translation may cause legal conflicts, it should be translated by manual to clarify the responsibility of the translation.

2.2 KBimCode-based Application

Taking advantage of BIM, the Korean government is establishing automated building permit system. As a part of the Korea government-led initiative for enhancing building design productivity as well as overall quality and performance using BIM, automated checking of Korea building permit requirements and rules is the motivation of this paper. The project can be summarized as follows: 1) development of an open BIM based quality verification system, 2) development of an open BIM based construction document optimization standard and applied technology, 3) development of an integrated cooperation work system during the public administrative process [7]. Several software tools that assist in the automated review process are also under development and will soon be publicly available [16]. Among the outcomes, this paper mainly focuses on the following development:

- KBimLogic

A logic rule-based management system that translates Korea Building Act (natural language) into computer-executable rule set files. Following the logic rule-based mechanism, the building code sentences are converted into an intermediate script language called KBimCode. KBimLogic includes 1) logic rule based-meta database and 2) KBimCode authoring tool. The logic rule-based meta database provides regulation-specific information that can be accessed by the KBimCode authoring tool and used for the generation and management of KBimCode.

- KBimCode

KBimCode is developed as a standardized rule set file that aims IFC-based open BIM. Design rules are translated into KBimCode and established as a database that can be reused and executed according to the user's purpose. KBimCode is generated and consistently managed by domain experts. Exported series of KBimCode can be used in various BIM assessment software such as KBimAssess-lite.

KBimCode-based application is a standardized process of rule representation, definition and evaluation. The process can be summarized as five stages.

1. Rules preparation (natural language),
2. Rule representation,
3. Rule definition,
4. Rule execution,
5. Result and report

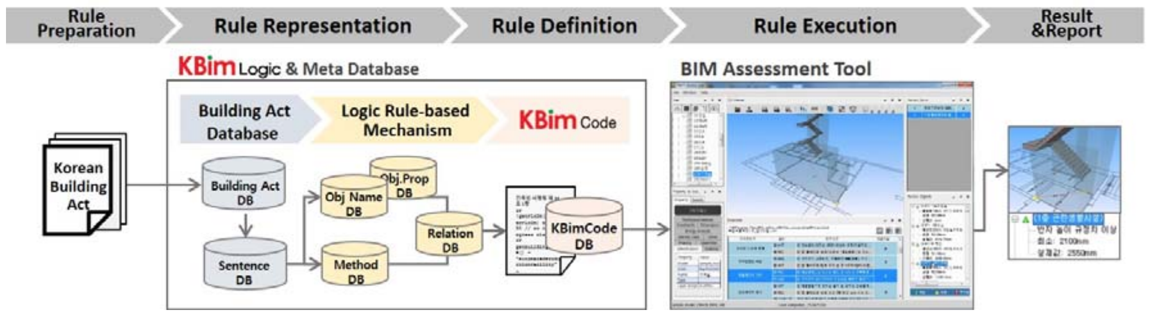


Figure 1. Overview of KBimCode-based applications and use of according software for each stage

Among the five stages of the KBimCode-based application process, this paper mainly deals with mechanism for rule representation, methods for rule definition, implementation issues with rule execution, and type of execution result and reports. Figure 1 illustrates overviews of KBimCode-based applications and use of according software for each stage.

3 Rule Representation

For the rule preparation, target rules are split into sentence unit and stored as a database. The database is used to author KBimCode through the logic rule-based mechanism. This mechanism provides a standardized way of rule representation [17, 18]. It can be explained in three parts according to the structure of a sentence: 1) noun phrases (objects/properties), 2) verb phrases (high-level methods), and 3) relation of sentences (Logic). The following sections describe each part in detail.

3.1 Noun Phrases: objects and properties

Noun phrases represent the key topic of a design rule. By classifying noun phrases, objects and associated properties handled in design assessment can be captured. Each object is developed in an object name database. Based on the name database, object and properties are classified as Figure 2.

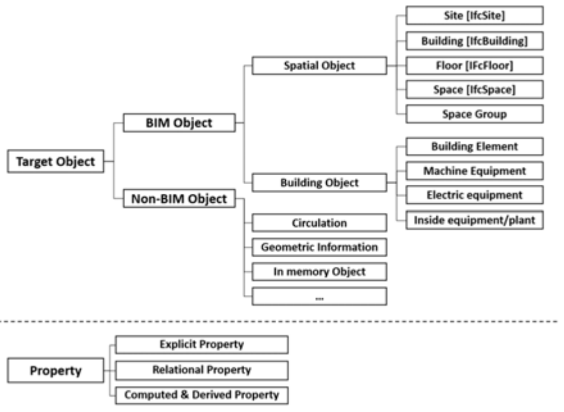


Figure 2. Classification of objects/property

Objects can be divided into two types: target object relevant to assessment of building design and non-target object that is exclusive to the assessment. The target object can be categorized as 1) BIM-enabled object (building objects and spatial object) and 2) non-BIM object derived from BIM data (circulation, geometric information, etc.). Properties are object-specific. They can be categorized into three types: 1) explicit property (e.g. object's name), 2) relational property (specified by two or more objects such as inclusion, distance, etc.), and 3) computed or derived property (e.g. building's gross area, floor number). Classification of objects and properties is mapped with standards such as IFC.

3.2 Verb Phrases: High-level Methods

Verb phrases define the main contents of assessment. Common rules obtained through a classification of verb phrase are defined as implementable methods. The methods are defined in high-level and distinguished from low-level methods for the implementation of BIM assessment tool. The high-level methods aims to represent verb phrases of natural language in intuitive form. The common rule become function and object/property become argument of the high-level methods. According to the object/property, the high-level methods can be categorized into four levels. At the first level, methods are distinguished into two types according to their checking target: 1) Object instance, and 2) property of the object's instance. In the second level, properties are categorized into four types: 1) basic property (object name, window type, etc.), 2) geometric property (door height, space area, etc.), 3) complex property (regulation-specific properties such as fireproof properties), and 4) relational property (inclusion, connectivity, distance, etc.). The third level specifies the content of checking. Contents of review are subdivided in this level. Lastly, methods are determined in the fourth level. There are two types of methods: 1) Representative methods that represent checking the types of the third level, and 2) extended methods subdivided from the representative methods. Extended methods check specific objects while the representative methods can review extensive objects. For example, a representative method 'getObjectArea()' has 'getFloorArea()', 'getWindowArea()', etc. as its extended methods.

Table 1 shows the example of high-level methods and their classification developed from a number of 1,977 Korean Building Act sentences related to building permit. Examples of methods on the fourth level are representative methods.

Table 1 Classification and example of high-level methods

1st Level (Target)	2nd Level (Property Type)	3rd Level (Checking Type)	4th Level (Method)
Object instance(s)		Object Query	`getObject()`
		Object Existence	`isExist()`
		Object Count	`getObjectCount()`
Property of Object	Basic Property	Property Query	`getProperty()`
		Object Type	`getObjectT`

instance(s)			`ype()`
		Object Material	`getObjectMaterial()`
		Object Usage	`getObjectUsage()`
	Geometric Property	Object Height	`getObjectHeight()`
		Object Width	`getObjectWidth()`
		Object Area	`getObjectArea()`
		Object Gradient	`getObjectGradient()`
	Derived Property	Material Type	`getObjectMaterialType()`
		Fire Safety	`isFireResistantCompartment()`
			…
	Relational Property	Inclusion	`hasObject()`
		Distance	`getObjectDistance()`
		connectivity	`isConnectedTo()`
			`isAdjacent()`
			…
		Path	`isAccessible()`
		Direction	`getObjectDirection()`

Specification of high-level methods is provided to software vendors for implementation. The high-level methods are mapped with low-level method library in BIM assessment tool. The detailed method specification for the example methods of Table 1 is available at [19].

3.3 Relation of Sentences: Logic

There are two types of relations pertaining to sentences: 1) relations in a sentence, and 2) relation between sentences.

3.3.1 Relation in a Sentence

Korean Building Act sentence consists of the checking condition and content [23]. Before checking the content, the condition should be satisfied. By clarifying the relation of the condition and content, logical composition of a sentence can be clarified. They are represented with IF-THEN-ELSEIF-ELSE logic. Figure 3 shows the sentence restructuring process of building permit rule to KBimCode. A sentence is split into semantic unit and their relations are figured. Based on the type of relationship, the sentence is restructured and represented as KBimCode.

The detailed description of the restructuring process is as follows:

1. Atomic sentence (AS)

Human can be aware of connected meanings of several sentences. However, computer cannot recognize the connected meaning of sentences in the same way as people do. Therefore, processing of split content of design rule with sentence-basis is needed. In this paper, atomic sentence is a minimum unit of sentence for assessment. It is a declarative sentence that returns either truth or falsity.

2. Translated atomic sentence (TAS)

Translated atomic sentence (TAS) is semantically split unit of atomic sentence. TAS has a subject (S) + verb (V) + object (Object) structure. TAS is obtained through the parsing of AS. With TAS, the checking condition and content of a sentence are clarified.

3. Arithmetic Logic Unit (ALU)

Originally, ALU is a digital electronic circuit that performs arithmetic and bitwise logical operations [20]. In this paper, ALU stands for a declarative clause that acts as the atomic unit for rule checking. The ALU is comprised of left operand, operator, and right operand [21]. The high-level methods and object/properties are placed in the left operand and compared with explicit values in the right operand using comparison operators.

4. Representation of rule (KBimCode)

Reflecting the logical composition of a sentence, ALU are restructured into KBimCode. The details of KBimCode syntax is handled in the next section. Table 2 shows the overall process of restructuring of natural language sentences to KBimCode. Enforcement Decree of Building Act (EDBA), article 64, clause 1 was used as an example

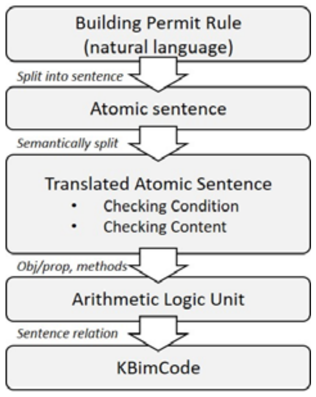


Figure 3. From sentence to KBimCode: Sentence restructuring process based on logic rule-based mechanism.

Table 2. Example of sentence restructuring process based on logic rule-based mechanism

Sentence	[Enforcement Decree of Building Act, article 64] A project owner of a building with six or more floors and a total floor area of 2,000 square meters or more shall have an elevator installed therein. In such cases, the size and structure of elevators shall be prescribed by Ordinance of the Ministry of Land, Infrastructure and Transport. [22]	
AS	A building with six or more floors and a total floor area of 2,000 square meters or more shall have an elevator installed.	
TAS	Condition	TAS1. A building has six or more floors TAS2. A building has a total floor area of 2,000 square meters or more
	Content	TAS3. A building has an elevator

ALU	Condition	ALU 1.
		`getBuildingStoriesCount() >= 6`
		ALU 2.
		`getGrossFloorArea() >= 2000`

	Content	ALU3. `isExist(Elevator) = TRUE`

KBim Code	`Check(BA_64_1){`
	`IF(getBuildingStoriesCount()>=6`
	`   AND getGrossFloorArea()>=2000)`
	`     THEN isExist(Elevator)=TRUE`
	`ENDIF`
	`}`	

3.3.2 Relation between sentences

Sentences may have relationships with other sentences. For instance, Korea Building Act sentences form complex cross referencing relationships as exemplified in Figure 4.

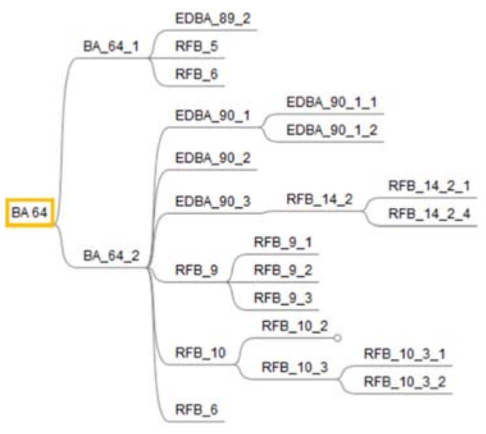


Figure 4. Example of relations between sentences. Each node stands for one sentence. The left-side nodes refer to the right-side nodes. For example, the sentences of the Building Act article 64, clause 1 (BA_64_1) refers to the sentences of the Enforcement Decree of Building Act article 89 clause 2 (EDBA_89_2), and the Regulations for Facility in Building article 5 and 6 (RFB_5, RFB_6).

Relationship between sentences are represented by control methods. For example, the Building Act article 64, clause 1 (BA_64_1) refers to the Enforcement Decree of Building Act 89, clause 2 (EDBA_89_2). Their relationship can be represented in the KBimCode of BA_64_1 as follows:

```
getResult(EDBA_89_2) = TRUE
```

Relations between sentences are developed into a database and directly used in KBimLogic application to logically combine sentences and generate set of KBimCode.

4 Rule Definition

4.1 KBimCode Language Design

This section deals with design strategy for KBimCode language definition. The design of KBimCode Language reflects features of logic rule-based mechanism. The strategy can be described with two points: 1) lexical design, and 2) syntactic design. The lexical design strategy plays as basic tokens and idioms in KBimCode Language. The subject (S) of TAS is specified in a dictionary and used as the fundamental lexicon. They are Korea Building Act-specific building objects and properties (KBimCode Object). The relations between objects and properties are represented as dot-notation. The verb (V) and object (O) structure will be used in a basic form of idioms. They are pre-defined as basic methods such as, getObectCount, getObjectHeight, isFireResistant, hasElement, getResult etc. More details of the predefined methods are depicted in [21].

The basic syntactic form of KBimCode is as follows:

```
check (arg) {
       Statement;
       …
}
```

The check is a pre-defined method that declares the checking of a certain building code sentence. The non-terminal token arg identifies a target building code sentence. The Statement is a rule defined in building code sentences. There are three types of statements: Arithmetic Logic Units (ALUs), Condition Statement, and KBimCode Object Model (KOM). ALUs is a single ALU or multiple ALUs joined with conjunctions such as

AND, OR. The ALUs is an expressive way of representing the rules. The condition statement is IF-THEN-ELSEIF-ELSE logic including ALUs as its statements. The KBimCode Object Model (KOM) is one of the key features of KBimCode Language. It enables users to define virtual objects with desired rules as constraints.

A set of statements can be grouped as statements group. The syntactic form of the statements group is as follows:

```
var {
   Statement;
   …
}
```

The statement group is defined outside of the basic syntactic form. The non-terminal token var is a user-defined variable name that represents the statements group. By using the statements group, KBimCode can be written in a clear and simplified way.

4.2 KBimCode Language Definition

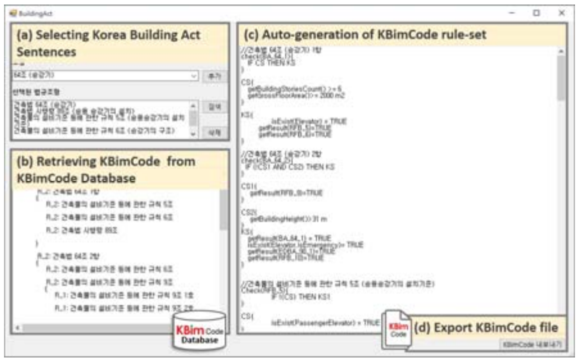
This section briefly introduces definition of KBimCode Language syntax and its grammar. The entire syntax of KBimCode is available at [24]. The formal definition described in this section is EBNF-based ANTLR rule [25]. ANTLR is a language parser generator and in its rule, non-terminals starting with lower case are syntactic rules and upper-case non-terminals are lexical definition.

Most of all, a non-terminal syntactic rule kKBimCodeProgram is the starting point of KBimCode Language definition. A lower case ‘k’ means KBimCode and the rest of the alphabets are simplified token to represent each syntactic component. As mentioned in the previous section, KBimCode Language basically has three key components 1) check declaration (the basic syntactic form), 2) statements group declaration, and 3) KBimCode statement. The check declaration and the statement group declaration are the syntactic form of KBimCode, while KBimCode Statement is used as components of the other two components. These three components are defined as: [2] kCheckDef, [3] kStatGruopDef, and [4] kStatDef. The snippet of the KBimCode definition is as follows:

[1] kKBimCodeProgram
```
kKBimCodeProgram
       :      kCheckDef? kStatObjDef?
       ;
```
[2] kCheckDef
```
kCheckDef
       :      CHECK ‘(‘ kCheckParamDef ‘)’
              ‘{‘ kStatDef ‘}’
       ;
```
[3] kStatObjDef
```
kStatObjDef
       :      kStatObject‘{‘ kStatDef ’}’
       ;
```
[4] kStatDef
```
kStatDef
       :      kStatLines
       ;
kStatLines
       :      kStatLine+
       ;
kStatLine
       :      (kALUsStat | kIfThenElseStat |
              kKOMDef)
       ;
```

4.3 KBimCode Database

The KBimCode database provides users with the means to acquire specific-purposed rules, even without knowledge of logic rule-based mechanism or KBimCode syntax. It is accessible through the KBimLogic application. As illustrated in Figure 4, there is an interface for selecting the target design rule (written in natural language). The rules are then logically organized based on pre-defined relationships between sentences. As a result, a KBimCode series can be exported by KBimLogic as a computer-executable rule set file such as JSON [26] and XML [27].


BuildingAct
(a) Selecting Korea Building Act Sentences
(b) Retrieving KBimCode from KBimCode Database
(c) Auto-generation of KBimCode rule-set
(d) Export KBimCode file
KBim Code Database
KBim Code

```
//[illegible] 64조 [illegible] 1항
check(BA_64_1){
   IF CS THEN KS
}
CS{
   getBuildingStoriesCount() >= 6
   getGrossFloorArea() >= 2000 m2
}
KS{
          isExist(Elevator) = TRUE
      getResult(RFB_5)=TRUE
      getResult(RFB_6)=TRUE
}
//[illegible] 64조 [illegible] 2항
check(BA_64_2){
   IF (CS1 AND CS2) THEN KS
}
CS1{
   getResult(RFB_9)=TRUE
}
CS2{
   getBuildingHeight() > 31 m
}
KS{
   getResult(KBA_64_1) = TRUE
   isExist(Elevator.isEmergency)= TRUE
   getResult(EDBA_90_1)=TRUE
   getResult(RFB_10)=TRUE
}
//[illegible]
Check(RFB_5){
          IF (CS) THEN KS1
}
CS{
          isExist(PassengerElevator) = TRUE
```

Figure 5. A screenshot of KBimLogic application: by selecting the target design rule (sentences), series of KBimCode can be automatically generated through KBimCode database.

Through the parsing library defined in the BIM assessment software (KBimAssess-lite), the rule set file is executed and compliance checking is performed. Example cases based on the Korean Building Act are tested and demonstrated in the next section.

5 Rule Evaluation

This section describes rule execution and evaluation based on actual demonstration. Specific information about the demonstration is as follows: 1) Design rule: Among Korean Building Act, regulations on the installation of an elevator (exemplified in the table 2) and the installation of escape stair were chosen for the demonstration. 2) BIM model: An IFC model of an actual office building in Seoul, Korea was used as a test model. This building is an office building and has a total floor area of 5,734 m2 with 4 basement floors and 15 ground floors. 3) BIM assessment software: KBimCode was developed to be software-independent and applicable in various BIM assessment tools. Specification of KBimCode is provided to software vendors for the development of syntax analyzer and parser. In this demonstration, one of BIM assessment tools named KBimAssess-lite was used.

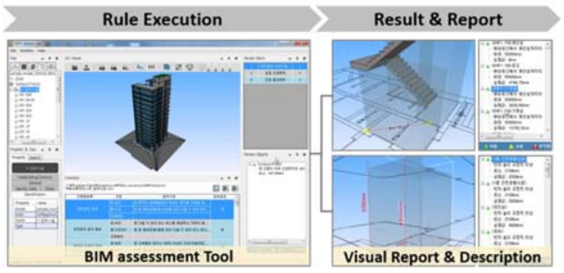


Figure 6. Overview of rule execution, result and report. Visualization as well as a description of the result is reported in KBimAssess-lite.

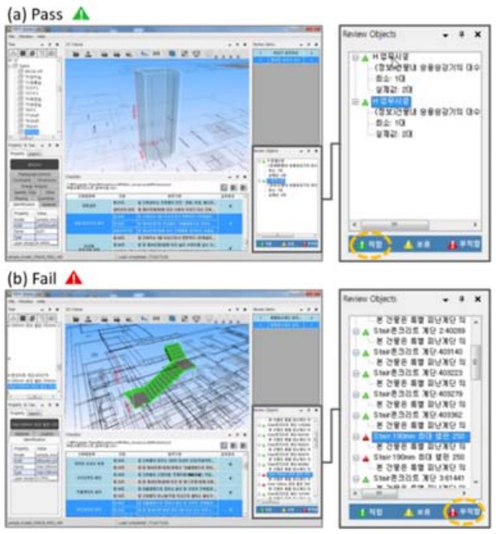


Figure 7. Example of rule execution result: pass case (a) and fail case (b).

KBimCode file is imported in the KBimAssess-lite and the rule is executed. Assessment result is provided with visual report and description. There can be three types of results: 1) pass, 2) fail, and 3) pending. In case of pending, the assessment result should be reviewed by domain expert in manual. Figure 7 shows detailed report of pass case and fail case. Visual report is given in the center of the interface and according description is provided in the lower right corner. The assessment result is reported in object unit. For example, in the (b) fail case of Figure 6, entire stairs in the target building is examined. Stairs with fail result is highlighted and provided with a detailed visual report and description of failing reason.

6 Conclusion

In this paper, we described KBimCode-based applications for rule representation, definition, and evaluation. For rule representation, Korean Building Act sentences decomposed into three parts: 1) noun phrase (object/property), 2) verb phrase (methods), 3) relation (logic). Following the logic rule-based mechanism, sentences are restructured into KBimCode. The definition of KBimCode was explored with two points: 1) lexical/syntactic design strategy, and 2) language definition with EBNF notation. KBimCode is developed into a database and reused for the architect's purpose. Evaluation of KBimcode was demonstrated with a BIM assessment tool named KBimAssess-lite. Result of rule execution can be three types: 1) pass, 2) fail, and 3) pending. Visual report as well as description is provided.

The KBimCode-based application provides a mechanism to generalize translation of natural language rules into a computable format for automated design assessment. In addition, it also suggests specific process for the application of KBimCode. KBimCode is an ongoing project and further verification is in progress. Its application and database will contribute to develop BIM-enabled quality assessment system for building permit in Korea. Although KBimCode was developed based on building permit requirements in Korea Building Act, its application can be extended to various design rules such as design guides, request for proposal etc. for automated code compliance checking.

Acknowledgement

This research was supported by a grant (15AUDP-C067809-03) from Architecture & Urban Development Research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

References

[1] J. Dimyadi, R. Amor, Automated Building code Compliance Checking Where is it at?, In Proceedings of CIB WBC 2013, Brisbane, Australia, pp. 172-185.

[2] J.Choi, J. Choi, I. Kim, Development of BIM-based evacuation regulation checking system for high-rise and complex buildings, Automation in Construction, 46(2014) 38-49, 2014.

[3] J-K. Lee, J. Lee, Y.-S. Jeong, H. Sheward, P. Sanguinetti, S. Abdelmohsen et al., Development of space database for automated building design review system, Automation in Construction, 24 (2012) 203-212.

[4] C.M. Eastman, P. Teicholz, R. Sacks, K. Liston, BIM Handbook—A guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors, John Wiley & Sons Inc., Hoboken, NJ, United States of America, 2008.

[5] C. Han, C. J. Kunz, and K. H. Law. Making Automated Building Code Checking a Reality, Facility Management Journal, 1997, pp. 22-28.

[6] D. Greenwood, S. Lockley, S. Malsane, J. Matthews, Automated compliance checking using building information models, In Proceeding of The Construction, Building and Real Estate Research Conference of the Royal Institution of Chartered Surveyors, Paris, France, 2010.

[7] I. Kim, Open BIM based Technological Environment for Building Design Quality Enhancement, Kyung Hee Univ., (Project report), 2015.

[8] W. Solihin, C. Eastman, Classification of rules for automated BIM rule checking development, Automation in Construction, 53(2015): 69–82.

[9] CORENET, CORENET Singapore, Available at: http://www.corenet.gov.sg/

[10] C. Eastman, J.-M. Lee, Y.-S. Jeong, J.-K. Lee, Automatic Rule-based Checking of Building Designs, Automation in Construction, 18(8):1011-1033, 2009.

[11] J-K. Lee, Automated checking of building requirements on circulation over a range of design phase, Ph.D. dissertation, Georgia Institute of Technology, 2012

[12] L. Ding, R. Drogemuller, M. Rosenman, D. Marchant, J. Gero, Automating code checking for building designs, in: K. Brown, K. Hampson, P. Brandon (Eds.), Clients Driving Construction Innovation: Moving Ideas into Practice, CRC for Construction Innovation, Brisbane, Australia, 2006, pp. 113–126.

[13] 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, Uhm et al., Advanced Engineering Informatics, 29(3): 602–615, 2015.

[14] J. Zhang, and N.M. El-Gohary, A.M.ASCE, Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking, Journal of Computing in Civil Engineering, 30(2), 2013.

[15] J. Zhang, S.M.ASCE1; and Nora M. El-Gohary, A.M.ASCE2, Automated information transformation for automated regulatory compliance checking in construction, Journal of Computing in Civil Engineering, 29, 2015.

[16] Open BIM based Technological Environment for Building Design Quality Enhancement, On-line: http://dq.kbims.or.kr/

[17] H. Lee, S. Lee, S. Park and J-K. Lee, An Approach to Translate Korea Building Act into Computer-readable Form for Automated Design Assessment, International Association for Automation and Robotics in Construction, 2015.

[18] H. Lee, J-K. Lee, S. Park and I. Kim, An Approach to Translate the Korean Building Act into a Computer-executable Form for Evaluating Building Permit Requirements, Automation in construction, in press.

[19] Method manual, Open BIM based Technological Environment for Building Design Quality Enhancement, On-line: http://designitlab.kr/bim/kbim/method/classification.asp

[20] Wikipedia, Arithmetic Logic Unit, On-line: https://en.wikipedia.org/wiki/Arithmetic_logic_unit

[21] S. Park, H. Lee, S-i. Lee, J. Shin, and J-K. Lee, Rule Checking Method-centered Approach to Represent Building Permit Requirements, International Association for Automation and Robotics in Construction, 2015.

[22] Korea Building Code (English), Statutes of Republic of Korea, On-line: http://elaw.klri.re.kr/kor_service/lawView.do?hseq=32490&lang=ENG

[23] H. Lee, S. Park, I. Kim, J-K. Lee, A Logical Rule-based Approach to the Korea Architecture Code Sentences for BIM-enabled Design Assessment Systems, Korea Design Knowledge Society, 2015.

[24] KBimCode Syntax, On-line: http://designitlab.kr/bim/kbim/kbimcode/KBimCodeLangDef.asp

[25] Parr: 2008, The Definition ANTLR Refernce: “Building Domain-Specific Languages", Pragmatic Bookshelf.

[26] JSON, ECMA-404 The JSON Data Interchange Standard, Online: http://www.ecmainternational.org/publications/files/ECMA-ST/ECMA-404.pdf

[27] XML, W3C, Online: http://www.w3.org/TR/WD-xml-961114.html