www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

DEFINITION OF A DOMAIN-SPECIFIC LANGUAGE FOR KOREAN BUILDING ACT SENTENCES AS AN EXPLICIT COMPUTABLE FORM

SUBMITTED: June 2016
REVISED: September 2016
PUBLISHED: November 2016 at http://www.itcon.org/2016/26
GUEST EDITORS: Dimyadi J. & Solihin W.

Seokyung Park,
Dept. of Interior Architecture Design, Hanyang University;
Seokyung.park529@gmail.com

Yong-Cheol Lee,
Department of Construction Management, Louisiana State University;
yclee@lsu.edu

Jin-Kook Lee,
Dept. of Interior Architecture Design, Hanyang University;
designit@hanyang.ac.kr

SUMMARY: *This paper aims to describe the definition of KBimCode Language and to demonstrate its actual use case. KBimCode is a domain-specific computer language to represent the regulatory sentences in the Korea Building Act as explicit computable rules currently focusing on the building permit-related requirements. As other domain-specific languages are usually in pursuit of both ease-of-use and adequate fidelity to deal with complex domain-specific issues, KBimCode also aims to accomplish a neutral and standardized way of rule-making in an easy-to-use syntax. To address how KBimCode has achieved such objectives, the following main topics are covered in this paper: 1) Language design: features of the Korea Building Act are reflected in a strategy for lexical and syntactic design of KBimcode Language; 2) Language definition: based on the context-free EBNF notation, specifications of KBimCode Language are described; 3) Demonstration: KBimCode can be applied to a BIM assessment tool and executed for rule checking. The examples presented in this paper describe the series of the process. KBimCode is one of the outcomes of an ongoing research project to develop automated design review systems for building permits in Korea.*

KEYWORDS: *Automated building permit system; Automated design assessment; Rule checking; Rule-making; Domain-specific language, KBimCode*

REFERENCE: *Seokyung Park, Yong-Cheol Lee, Jin-Kook Lee (2016). Definition of a domain-specific language for Korean building act sentences as an explicit computable form. Journal of Information Technology in Construction (ITcon), Special issue: CIB W78 2015 Special track on Compliance Checking, Vol. 21, pg. 422-433, http://www.itcon.org/2016/26*

1. INTRODUCTION

Code compliance checking of building design is an important task that has to be taken into consideration in the entire design process. In a conventional process, code compliance is manually checked by architects or rule experts in an iterative, time-consuming, and error-prone manner (Ding, 2006; Greenwood, 2010). As Building Information Modeling (BIM) is being incorporated into an increasing number of fields within architecture, engineering, and the construction and facilities management (AEC-FM) industries, BIM-enabled code compliance checking is explored as a promising direction of BIM application (Lee, 2012; Eastman 2009a). BIM-enabled design assessment is expected to support a conventional assessment process and increase efficiency and precision (Han, 1997; Eastman, 2009b). Taking advantage of BIM-enabled and automated design assessment, the Korean government is on its way to enhancing the quality of building designs. As part of the effort, a government-funded research and development project for developing a BIM-enabled code compliance checking system for building permits is the basis of this paper.

Related research and development in automated code compliance checking have explored the interpretation of rules as a critical task to accomplish error-free results. Design rules are written in human language (natural language), which inevitably includes ambiguity, vagueness and has limitations (Nawari, 2012). Therefore, natural language rules are formalized into explicit representations and translated into computational rules to enable automatic reasoning (Malsane, 2015). In early attempts to develop computable rules in the field of AEC industry, Fenves (Fenves, 1966) formalized specifications of the American Institute of Steel Construction (AISC) using a decision table from the 1960s. Recent research on BIM-based design assessment of buildings has increased since 2007 (Yalcinkaya and Singh, 2016) with emphasis various methods. CORENET (COnstruction and Real Estate NETwork) of Singapore, which is an early initiative of BIM-enabled code compliance checking, translated the Building Code of Singapore by a hardcoded method (Teo and Cheng, 2006). DesignCheck of Australia formalized the Australian Standard (AS) 1428.1 to pre-implementation specification and encoded it with EXPRESS language (Ding, 2006). For automated checking of circulation and security requirements, GSA courthouse design guide was parameterized and implemented on top of the Solibri Model Checker platform using JAVA language (Eastman, 2009b). Uhm et al analysed request for proposal (RFP) and encoded design rules with SWRL language (Uhm, 2015). Zhang et al (Zhang, 2013; 2015) devised algorithms that automatically extract and transform information using a natural language processing (NLP) approach.

Existing studies showed the possibility of various rule-making approaches and their application to automated code compliance checking. However, current rule-making approaches require extensive programming knowledge or expertise to generate computable rules. In addition, there remain limitations in directly applying these existing efforts to the translation of the Korean Building Act due to its locality and distinct characteristics. The following features should be carefully taken into consideration when establishing a translating strategy for the Korean Building Act.

1. Complex relation: Korean Building Act sentences are commonly cross-referenced. In addition, the display of relationships is inconsistent; i.e., the relations are marked randomly in referencing or referenced sentences. Such features increase the complexity, which in turn increase discrepancies in the interpreted building act sentences.

2. Integrity: The Korean Building Act carries legal binding force. Interpretation of building act sentences should assure integrity without errors. Errors in interpretation might lead to unnecessary conflicts. Therefore, it is important to assure the integrity of the interpretation and make it clear which party is responsible for possible errors.

As one of rule-making solutions for the Korea Building Act, domain-specific language (DSL) called KBimCode was introduced (Lee et al. 2016). It has been developed as a way to translate building permit-related regulations in the Korea Building Act into computable rules reflecting distinct features of Korean Building Act sentences. Regarding the two features of the Korea Building Act, KBimCode 1) represents Korea Building Act sentences with explicit relations, and 2) ensures the integrity of legal interpretation since rule experts author and manage them. KBimCode aims to be intermediate and independent from specific BIM environment or proprietary software. It is processed in open text code and can be directly used in various BIM assessment tool. The specification of KBimCode is provided to software vendors to make KBimCode executable in target assessment tools. As one of specification that enables such implementation, this paper aims to describe language design and

definition. The following sections introduce 1) an overview of KBimCode, 2) a strategy for language design, 3) specifications of KBimCode based on context-free EBNF notation and associated texts, and 4) language evaluation.

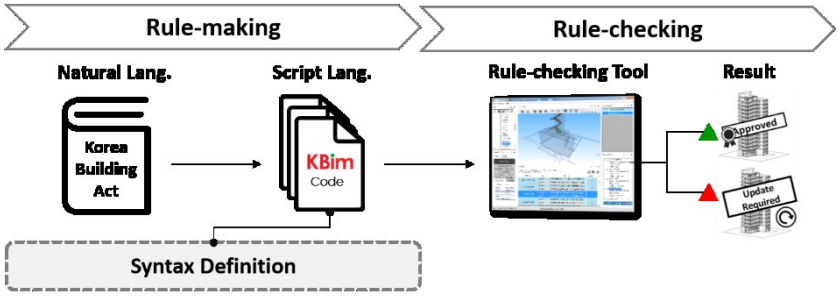


FIG. 1: Korea Building Act sentences are translated into domain-specific language called KBimCode and then applied to rule checking. This paper aims to define syntax of the KBimCode.

2. KEY FEATURES OF KBIMCODE

A distinguishing feature of KBimCode is that it is a standardized and software-independent approach to the rule-making process. Previous projects embedded design knowledge into rule checking tools. In such cases, the rule-making process was integrated into the development of rule checking software. The resultant rules are inevitably dependent on specific proprietary software and a closed BIM environment. Conversely, KBimCode is an intermediate script language that is independent from rule checking tools. KBimCode is processed into open text files such as JSON (www.ecma-international.org: Nov 2015), XML (www.w3.org: 2015) or other formats and can be used in various BIM assessment tools. KBimCode is generated through a standardized process that is described elsewhere (Lee, 2015).

Based on this overview, KBimCode can be summarized as follows:

1) KBimCode can be classified along high-level computer language categories as; 1. a domain-specific language, 2. a programming language, and 3. a rule language.

2) Its main functionality is to represent natural language rules as computable rules for automated code compliance checking.

3) Its target field is architecture, engineering, construction and facility management (AEC-FM) industry.

4) KBimCode Language attempts to process the Korean Building Act in an intuitive way in order to define user-friendly computable rules.

5) It does not directly modify and change building models because it aims to only assess building design models.

The main targets of KBimCode are the building permit-related regulations in the Korean Building Act. Building permit regulations in their entirety were analyzed to capture the general syntactic and lexical elements for language design. For example, specific building objects in the Korean Building Act and their properties were clarified for use as the lexicon of KBimCode. Moreover, various expressions were devised in advance to represent specific conditions of building act sentences. For instance, we adopted dot notation approach to represent building objects' properties. We also predefined methods to represent relations between sentences and assessment of iterative rules. (Park et al, 2015, Lee et al, 2015).

3. KBIMCODE LANGUAGE DESIGN APPROACH

3.1 Lexical Design

This section describes a lexical design strategy for applying basic tokens and idioms to the KBimCode Language. The fundamental strategy is outlined as follows:

1) The Korean Building Act is divided into atomic sentences (ASs), a type of declarative sentence that is either true or false. It is the minimum unit of rule checking. AS are processed into translated atomic sentence (TAS) that have simplified S (subject) + V (verb) + O (object) structures.

2) KBimCode assumes that S are building objects and properties specific to the Korea Building Act-specific. They are compiled in a dictionary and used as the fundamental lexicon.

3) Dot notation is used to represent properties and relations of the objects.

4) Predefined object names start with upper case letters while property names and user-defined names start with lower case.

5) VO structure is used in a basic form of idioms.

6) VO structure can be pre-defined as the following methods: getObect, getObjectProperty, isExist, getDistance, etc. The details of the predefined methods are described in (Park et al, 2015).

7) To instantiate checking of a particular building act sentence: check(argument)

3.2 Syntactic Design

This section briefly previews a syntactic design style for the KBimCode Language. A Basic syntactic form of KBimCode can be represented as follows:

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

- Where check is a pre-defined method that declares the checking of a certain building act sentence.

- Where “check”, “arg” and “Statement” are non-terminal tokens.

The non-terminal token `check` is a pre-defined control method that declares the checking of certain building act sentence pointed by `arg`. The `Statement` represents rules defined in building act sentences. The type of statemenst can be categorized as follows;

- KBimCode Object Model (KOM)
- Conditional Statement
- Arithmetic Logic Units (ALUs)

The KBimCode Object Model (KOM) is a key feature of the KBimCode Language. It is a human-centeric abstraction of building objects specified in the Korean Building Act. KOM is a borrowed concept from the BERA Object Model (BOM) (Lee, 2011). By using KOM, a user can define virtual objects with desired rules and with constraints. More details about KOM are in the next section.

A Korean Building Act sentence semantically consists of two parts: a checking condition and checking content. The logical relationship between the two components is expressed with IF-THEN-ELSEIF-ELSE logic. The checking conditions and checking content are represented as ALU. Originally, ALU is a digital electronic circuit that performs arithmetic and bitwise logical operations (ALU, 2016). In this paper, ALU stands for a declarative clause that performs as the atomic unit for rule checking. The composition of ALU is detailed in the next section.

A set of statements can be grouped as a Statement Group. The syntactic form of the Statements Group is as follows;

```
var {
        Statement;
        …
}
```

- Where var and Statement are non-terminal tokens.

A Statements Group is declared outside of the basic syntactic form and its name is used as a variable inside the form. By using Statements Groups, KBimCode can be written in a clear and simplified way. Moreover, Statements Groups and KOM provide reusable variable names that prevent users from writing iterative code.

4. FORMAL DEFINITION OF KBIMCODE

4.1 Check Declaration and Statement Group

This section defines the syntax of KBimCode language grammar. The formal definition described in this section is EBNF-based ANTLR rule (Parr, 2008). ANTLR is a language parser generator. Non-terminals starting with lower case are syntactic rules and upper-case non-terminals are lexical definitions. This section shows snippets of \KBimCode definition. The entire EBNF can be found in * KBimCode Syntax Manual (KBimCode Syntax Manual, 2016).

As mentioned in the previous section, the KBimCode language has three key components: 1) A check declaration, 2) A statements group, and 3) A KBimCode statement. The check declaration and the statement group make up the syntactic form of KBimCode, while KBimCode statements are subordinate to the other two components. These three components are defined as: [1] kCheckDef, [2] kStatGruopDef, and [3] kStatDef. Most of all, a non-terminal syntactic rule kKBimcodeProgram is the starting point of KBimCode language definition. A lower case 'k' refers to KBimCode and other letters of the alphabet are simplified tokens representing each syntactic components.

```
kKBimCodeProgram
        :       kCheckDef kStatObjDef*
        ;
```

[1] Definition: kCheckDef

```
kCheckDef
        :       CHECK '(' kCheckParamDef ')'
                '{' kStatDef '}'
        ;
```

Check Declaration is an essential component of KBimCode. In its syntactic rules kCheckDef, the lexical rule, CHECK stands for the pre-defined method "check" that instantiates checking of a target building act sentence. KCheckParamDef defines identifier tokens that represent the target sentence.

[2]Definition: kStatObjDef

```
kStatObjDef
        :       kStatObject '{' kStatDef '}'
;
```

A Statement Group is optional. It is useful to represent code in an explicit way. The syntactic rule kStatObject defines a user-defined variable as the name of the Statement Group.

*Entire EBNF-based KBimCode definition is described in KBimCode Syntax Manual, available at http://designitlab.kr/bim/KBimCodeSyntaxManual.asp

4.2 KBimCode Statement

There are three types of KBimCode statements: KOM, Condition statement, and ALUs. These three types are defined as [3-1] kKOMDef, [3-2] kIfThenElseStat, and [3-3] kALUsStat. The definition of a KBimCode statement and each type of statement are as follows;

[3]Definition: kStatDef

```
kStatDef
        :       kStatLines
        ;
kStatLines
        :       kStatLine+
        ;
kStatLine
        :       (kKOMDef | kIfThenElseStat | kALUsStat)
        ;
```

4.2.1 KBimCode Object Model

The KBimCode Object Model is an abstracted state of the building model. It is based on Korean Building Act-specific objects and properties. There are two types of KOM: static KOM and dynamic KOM. Static KOM is a static data set already defined in a given building model. Therefore, all given attributes can be established statistically when loading the building model. The dynamic KOM is a user-defined subcollection of static BOM. It is dynamically instantiable with user-defined constraints. By using KOM, the objects of interest are derived from the model.

KOM is defined by a basic syntactic form as follows:

```
ObjectType var {
        Statement
        …
}
```

- Where “ObjectType”, “var”, “Statement” are non-terminal tokens.

The non-terminal token `objectType` is part of the KOM lexicon based on the KBimCode Object dictionary. `Var` is a user-defined variable name. The dynamically instantiable KBimCode Objects can be replaced with user-defined names. `Statement` are KBimCode statements.

The definition of KOM is as follows:

[3-3] KOM definition: kKOMDef

```
kKOMDef
        :       kKOMDefStat
        ;
kKOMDefStat
        :       kKOMDefStatDec
                '{' kStatDef '}'
        ;
kKOMDefStatDec
        :       bWrapObjType BID
        ;
```

The `kKOMDefStatDec` defines building objects and user-defined variable name. The `bWrapObjType` determines the type of building object. The lexical rule `BID` can be instantiated by any of the variable names.

The following is an example with actual building design rule defined in the Korean Building Act (Enforcement Decree of Building Act, Article 35, Clause 1) (Korean Building Act: 2016)

"… Cases where the total floor area of the fifth or upper floor is 200 or fewer square meters…"

This spells out two rules regarding the floor in certain constraints. The divided sentences and following KBimCode segments are as follows:

S1) Floor number is fifth or higher

S2) The total floor area of the floors is 200 or fewer square meters

K1)
```
Floor myFloor {
                Floor.number >= 5
        }
```

K2)
```
getFloorArea(all.Floor) <= 200
```

In the example K1), the user variable myFloor contains all the floor objects that are on the fifth floor or higher. This is an example of user-defined object group (dynamic KOM), as a collection of floors. KOM is used together with ALUs or condition statements to perform rule checking as shown in KBimCode segment K2).

4.2.2 Condition Statement

[3-2] Condition Statement definition: kIfThenElseStat

```
kIfThenElseStat
        :       kIfStat         kThenStat  kElseIfStat*  kElseStat*
        ;
kIfStat
        :       IF '(' kALUsStat')'
        ;
kThenStat
        :       THEN kALUsStat
        ;
kElseIfStat
        :       ELSEIF '(' kALUsStat')' kThenStat
        ;
kElseStat
        :       ELSE kALUsStat
        ;
```

The conditional statement inheres IF-THEN-ELSEIF-ELSE logic. ELSEIF and ELSE are optional. The following are example snippets:

```
1) IF (BuildingStoriesCount() >= 6) THEN isExist(Elevator) = TRUE
2) IF (CS) THEN KS
```

Logically, 1) and 2) are same. The inherent ALUs can be replaced with variable name of the Statements Group (in this example, CS replaces "BuildingStoriesCount() >= 6" and KS replaces "isExist(Elevator) = TRUE").

4.2.3 Arithmetic Logic Unit (ALUs)

An Arithmetic Logic Unit is an atomic rule for checking. It returns true or false results by comparing left operands and right operands with operators. A left operand describes a specific condition while a right operand states an explicit value. With AND, OR conjunctions, multiple ALUs can be expressively joinedThe definition of ALUs is as follows:

[3-1] ALUs definition: kALUsStat

```
kALUsStat
       :      kALUStat ((AND | OR) kALUStat)*
       ;
kALUStat
       :      kOperand kALUOperator kOperand
       ;
kOperand
       :      kLRMethod
       |      kCtrlMethod
       |      kKOMGetterByPropC
       |      kKOMGetterByBIDC
       |      BID
       |      '"' BID '"'
       |      BOOLEAN
       |      INTLITERAL
       |      DOUBLELITERAL
       ;
```

The `kALUsStat` defines multiple ALUs joined with conjunctions. The `kALUStat` defines a single ALU. The `kOperands` can be predefined methods (kLRMethod, kCtrlMethod), dot notation access to building objects and properties (kKOMGetterByPropC, kKOMGetterByBIDC), or explicit values such as truth or false, string, or numeric values (`BID`, `BOOLEAN`, `INTLITERAL`, `DOUBLELITERAL`). Operators are data type-specific. Operators defined by the `kALUOperator` can be as follows:

- Operators for string: =, !=, ==
- Operators for numeric: >, >=, =, <, <=, !=

Some examples of valid KBimCode segments within the `kALUsStat` rule are as follows:

```
1) getTotalFloorArea(myFloor) >= 200m2
2) Building.Usage = "MultiUnitHouse" AND getBuildingStoriesCount() >= 6
3) getResult(REFB_16_1) = TURE
```

5. DEMONSTRATION

Previous sections described design strategy and definition methods of KBimCode syntax. This section takes a step forward and demonstrates a use-case of KBimCode Language. Evaluating fidelity of the language is not the point of this section. Currently, over 1,100 Korea Building Act sentences have been translated into KBimCode with the current versions of syntax definition. These operate as actual codes and can be utilized in rule-checking. For the demonstration, a series of Korea Building Act sentences were shown as examples. These were 1) translated into KBimCode, and 2) executed in a BIM assessment tool. For the rule execution, KBimAssess-lite, a rule-based BIM model checking program for conformance checking of the Korean Building Act was used as a target BIM assessment tool.

5.1 An Example Regulatory Sentence and its KBimCode

Natural language sentences are translated into KBimCode according to 'logic rule-based mechanism'. The mechanism is elaborated in Lee et al. 2016. Once sentences are translated to KBimCode, they are developed in a database as a reusable source for architects. We selected sample Korean Building Act sentences and retrieved a series of KBimCode from the database. The sample building act sentences regulate requirements for the installation of a living room ceiling according to the Enforcement Decree of the Korean Building Act, article 50 (EDBA_50) and Regulation for Egress and Fireproof Construction of Building, article 16, clause 1 (REFB_16_1) (Korean Building Act, 2016). They are associated by their detailed contents. Table 1 shows the building act sentences and the corresponding KBimCode. The grammar of the KBimCode was validated as a parsing tree, auto-generated by a KBimCode syntax analyzer, proved.

TABLE 1. Example of KBimCode based on the actual Korean Building Act: "Enforcement Decree of Building Act Article 50"

Korean Building Act Sentences (Natural Language)

Enforcement Decree of Building Act, article 50 (EDBA_50)
"Direct stairs installed on the fifth or upper floor or the second or lower underground floor shall be installed as fire escape stairs or special escape stairs: Provided, That the same shall not apply to cases where main structural parts are made of a fireproof structure or non-combustible materials and falls under any of the following subparagraphs:"

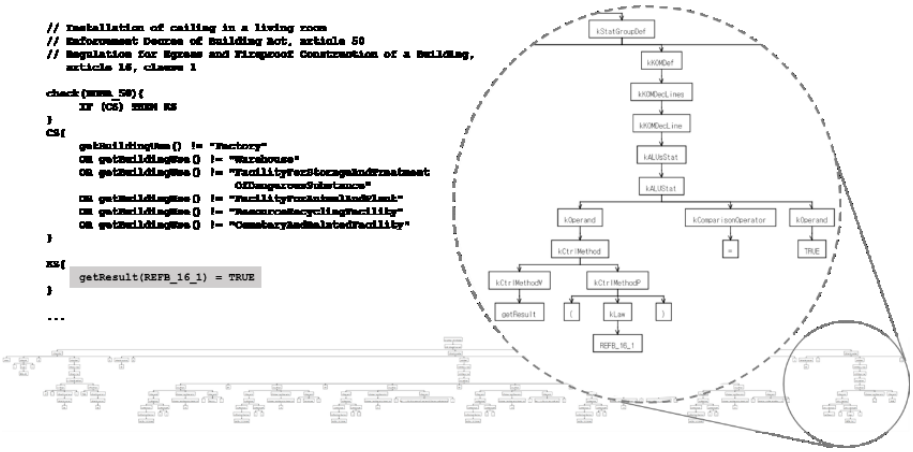
Regulation for Egress and Fireproof Construction of Building, article 16, clause 1 (REFB_16_1)
"According to the Enforcement Decree of Building Act, article 50, ceiling installed in a living room shall be the height of more than 2.1 meters;"

KBimCode

```
// Enforcement Decree of Building Act, article 50
check(EDBA_50){
    IF (CS) THEN KS
}
CS{
    getBuildingUse() != "Factory"
    OR getBuildingUse() != "Warehouse"
    OR getBuildingUse() != "FacilityForStorageAndTreatmentOfDangerousSubstance"
    OR getBuildingUse() != "FacilityForAnimalAndPlant"
    OR getBuildingUse() != "ResourceRecyclingFacility"
    OR getBuildingUse() != "CemeteryAndRelatedFacility"
}
KS{
    getResult(REFB_16_1) = TRUE
}

// Regulation for Egress and Fireproof Construction of Building, article 16, clause 1
check(REFB_16_1){
       getSpaceHeight(LivingRoom, b) >= 2.1m
}
```

Grammar Validation (Parsing Tree)



* The highlighted code is equivalent to the enlarged image.

5.2 An Actual Use Scenario of KBimCode

KBimCode was developed to be an independent and non-proprietary language that can be executed in various BIM assessment tools. Specification of KBimcode, such as EBNF-based KBimCode language definition, a dictionary of regulation-specific objects and their properties, a specification of implementable methods etc. are provided to software vendors for the development of KBimCode translators. A translator maps KBimCode and proprietary rule-sets of target BIM assessment tools.

FIG 2 illustrates the mapping between KBimCode and KBimAssess-lite based on the KBimCode translator developed from the given specification. The rule-set of KBimAssess-lite is relatively longer than KBimCode, because it is iterative. For example, checking the height of a living room can be represented with a single line of code using KBimCode. However, the rule-set of KBimAssess-lite includes various implementation issues including low-level methods for querying, mapping, and calculating IFC objects and their properties.

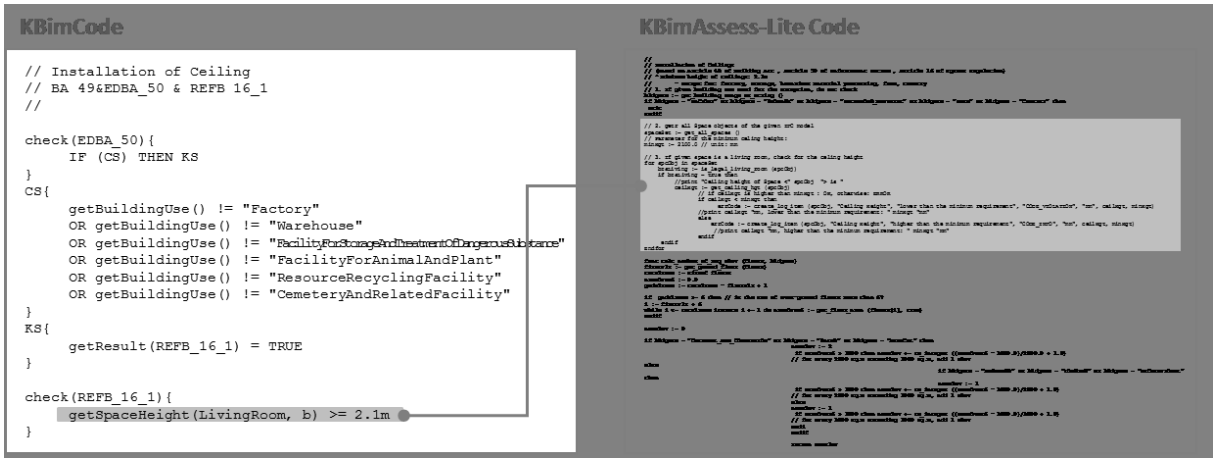


FIG. 2: Example of a comparison between KBimCode (left) and a language of target program (right). In this case, KBimAssess-lite was used as a target program. KBimCode is declarative and comparatively shorter than the target language of KBimAssess-lite. The shaded box handles semantically same contents.

FIG 3 shows use scenario of KBimCode in three steps: 1) translation of Korean Building Act sentences (natural language) to KBimCode and its file, 2) Importing KBimCode file to a BIM assessment tool (KBimAssess-lite), and 3) Execution of KBimCode and result. Once Korean Building Act sentences are translated into KBimCode, they are developed to database for reusability, Desirable KBimCode is extracted from the database as a form of computer-executable rule set file such as Json, XML etc. The file is then imported into a BIM model checker and executed. Currently, KBimAssess-lite provides results of rule checking in both visual and textual reports. FIG 3 shows the snapshot of the example KBimCode represented in Table1.

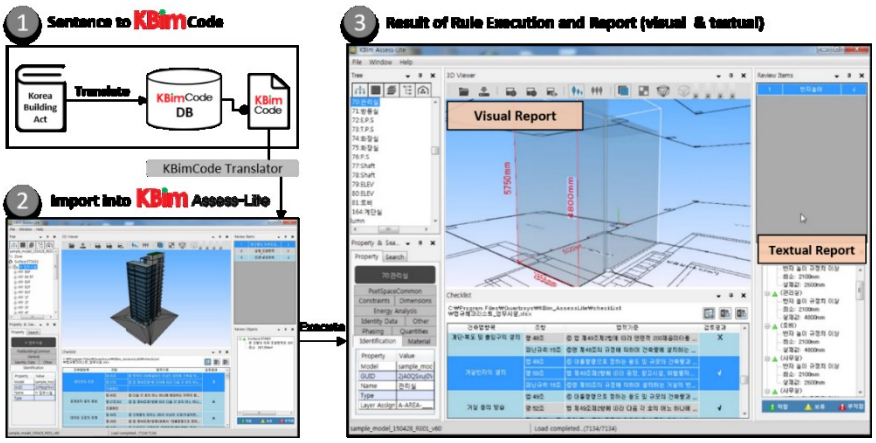


FIG. 3: Use scenario of KBimCode with actual KBimCode example.

6. CONCLUSION

This paper described the approach to designing and defining KBimCode. The lexical and syntactic form of the KBimCode Language reflects features of building permit regulations in the Korean Building Act and can be categorized as: 1) Check Declaration, 2) Statement Group, and 3) KBimCode Statements. The Check Declaration points to a target building act sentence and contains series of statements. The type of statements is ALUs, conditional statements, and KOM. Specifically, KOM is a key feature of KBimCode. By using KOM, users can specify objects of their own interest. The series of statements can be grouped as a Statement Group, which is a reusable object. We also demonstrated KBimCode of actual regulatory sentence KBimCode with a BIM assessment tool named KBimAssess-lite. With the specification of KBimCode, a translator can be developed and KBimCode can be executed in a target BIM assessment tool. KBimCode development is an ongoing project. Limitations exist in the current version of KBimCode and syntax update, modification, improvement etc. are being made. Further development will reflect extended target sentences. In addition, targets of KBimCode will extend to various design rules such as design guidelines, requests for proposals, etc.

ACKNOWLEDGEMENT

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

This paper was partially based on a conference paper presented in Computer-Aided Architectural Design Research in Asia (CAADRIA) 2016, Melbourne.

REFERENCES

Lee J.-K., Lee J., Jeong Y.-S., Sheward H., Sanguinetti P., Abdelmohsen S. and Eastman C.M. (2012). Development of space database for automated building design review systems, *Automation in Construction*, Vol. 24, 203–212.

Eastman C.M. (2009a), Automated assessment of early concept design, *Article in Architectural Design Special Issue: Closing the Gap*, Vol. 79, Issue 2, 52–57.

Ding L., Drogemuller R., Rosenman M., Marchant D. and Gero J. (2006). Automating code checking for building designs, *Proceeding of Clients Driving Innovation: Moving Ideas into Practice*, Brisbane, Australia, 1-16.

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

Han C.S., Kunz J. and Law K.H. (1997). Making automated building act checking a reality, Facility Management Journal, 22–28.

Eastman C.M., Lee J., Jeong Y.-S., Lee J.-K. (2009b). Automatic Rule-based Checking of Building Designs, *Automation in Construction*, Vol. 18, Issue 8, 1011-1033.

Nawari O. N. (2012). Automated Code Checking in BIM Environment, *Proceeding of 14th International Conference on Computing in Civil and Building Engineering*, Moscow.

Yalcinkaya M. and Singh V. (2016). Patterns and trends in Building Information Modeling (BIM) research: A Latent Semantic Analysis, *Automation in Construction*, Vol. 59, 68–80.

Malsane S., Matthews J., Lockley S., Love P.E.D. and Greenwood D. (2015). Development of an object model for automated compliance checking, *Automation in Construction*, Vol. 49, Part A, 51–58.

Fenves S.J. (1966). Tabular decision logic for structural design, *Journal of the Structural Division,* Vol. 92, Issue 6, 473–490.

Uhm M., Lee G., Park Y., Kim S., Jung J. and Lee J.-K. (2015). Requirements for computational rule checking of requests for proposals (RFPs) for building designs in South Korea, *Advanced Engineering Informatics*, Vol. 29, Issue 3, 602–615.

Zhang J. and El-Gohary N. (2016). Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking, *Journal of Computing in Civil Engineering*, Vol. 30, Issue 2.

Zhang, J. and El-Gohary, N. (2015). Automated information transformation for automated regulatory compliance checking in construction, *Journal of Computing in Civil Engineering*, Vol. 29.

ECMA international (2013). The JSON Data Interchange Format, Retrieved from http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (Last Accessed: Nov, 2015).

W3C (2014). Extensible Markup Language (XML), Retrieved from http://www.w3.org/TR/WD-xml-961114.html (Last Accessed: Nov, 2015).

Lee H., Lee S., Park S., and Lee J.-K. (2015). An Approach to Translate Korea Building Act into Computer-readable Form for Automated Design Assessment, *Proceeding of ISARC2015*, University of Oulu.

Park S., Lee H., Lee S., Shin J. and Lee J.-K. (2015), Rule Checking Method-centered Approach to Represent Building Permit Requirements, *Proceeding of ISARC 2015*, Oulu.

Lee H., Lee J.-K., Park S and Kim I. (2016). Translating building legislation into a computer-executable format for evaluating building permit requirements, *Automation in Construction,* Vol. 71, 49-61.

Wikipedia, Retrieved from https://en.wikipedia.org/wiki/Arithmetic_logic_unit, (Last Accessed: Nov, 2015).

Parr T. (2008). The definitive ANTLR Reference: Building Domain-Specific Languages, Pragmatic Bookshelf,

Open BIM based Technological Environment for Building Design Quality Enhancement (2015), KBimCode Syntax Manual, Retrieved from http://designitlab.kr/bim/KBimCodeSyntaxManual.asp (Last Accessed: Nov.2015).

Lee J.-K. (2011), Building Environment Rule and Analysis (BERA) Language, *Ph.D. Dissertation*, Georgia Institute of Technology.

National Statute Data Center Korea, Korean Building Act, http://elaw.klri.re.kr/kor_service/main.do (Last Accessed: Nov 2015).

Park S. and Lee J.-K., Definition of a Domain-specific Language to Represent Korea Building ACT Sentences as an Explicit Computable Form, *Proceeding of CAADRIA 2016*.