İzmir Municipality Housing and Zoning Code Analysis and Representation for Compliance Checking

Sibel Macit, M. Emre İlal, H. Murat Günaydın
İzmir Institute of Technology, Turkey
Georg Suter,
Vienna University of Technology, Austria
sibelmacit@iyte.edu.tr

Abstract. Systems for code compliance checking of building projects require representation of building codes. Building codes are complex, and the development of computer implementable representations is challenging. As a case in point, this paper reports on experiences gained while modeling İzmir Municipality Housing and Zoning Code (IMHZcode). First, IMHZcode was analysed to understand the various types of information contained in it in order to develop a comprehensive building code model. The rules were classified according to their formalizability and self-containedness. Then, existing modeling approaches were evaluated to find the most convenient method that meets the needs for modeling IMHZcode. A key criterion used in this evaluation was ease of maintenance by non-programmers. The paper concludes with an illustrative example of the selected methodology's application within the context of IMHZcode.

1. Introduction

Development of code compliance checking systems has been an area of research that aims to provide computational support for accurate compliance checking of building projects against applicable building codes in a time and cost effective way. Research in this area has focused mostly on representation of building codes in computational format, definition of building model views, compliance checking algorithms and reporting.

The research presented in this paper focuses on computer-based representations of building codes for automated compliance checking. Although there has been much interest in this subject, impact on the AEC industry has been limited. A literature survey suggests two main reasons. One reason is that most of the previous building code models were not sufficiently comprehensive and too simplistic compared to the complex nature of building codes. To be comprehensive, a building code model should be able to represent all of the various types of rules in building codes. The second reason is that previous building code models are difficult to maintain. Building codes change continuously and the model should be able to accommodate addition of new rules and modification of existing ones. Non-programmer code authors should be able to easily carry out such model updates. In order to avoid these two shortcomings in previous efforts, current work focuses on two issues.

The first is to determine different types of rules by analyzing building codes so that a building code model can be defined correctly and comprehensively. İzmir Municipality Housing and Zoning Code (IMHZcode) has been chosen for this study, as it is representative of codes that are in effect across Turkey. IMHZcode was analyzed to understand the characteristics of building codes in these following steps: 1) Examining the organization of building code sections, 2) Identifying the basic elements of building codes, and 3) Classifying the rules in the building code according to their formalizability and self-containedness.

The second issue is the development of a building code model based on a representation methodology that accommodates a high level of maintainability. An investigation of how far existing modeling approaches meet the needs for modeling IMHZcode has been carried out. Although there has been several building code modeling studies in literature (Fenves, 1966; Kiliccote et al., 1994; Han et al., 2002), there is still no widely used formal methodology to represent rules in computer implementable format. The most common solution is to hard code rules into the compliance checking applications. This approach can be practical for implementing small number of simple rules but it is not generalizable and maintainable. The SMARTcodes (AEC3, 2012) approach, in contrast, aims to enable non-programmers to define computable rules using simple tools. It is based on the RASE (Requirement, Applicability, Selection, Exception) methodology where rules are broken down into four constructs. Non-programmers can identify and markup these four constructs in the actual text of the code.

At the end of the study, IMHZcode rules pertinent to residential buildings have been represented in computer implementable format based on the RASE method.

2. Building Code Analysis

A building code is a legal document that specifies the minimum conditions for a certain aspect of a building construction. The main purpose of building codes is to protect public health, safety and general welfare as they relate to the construction and occupancy of buildings and structures. Building codes are determined by appropriate authorities in different domains and may vary widely from country to country.

In Turkey, all legal arrangements concerning construction fall under the responsibility of the Ministry of Public Works and Settlement. There are two laws in force: Construction Law No.3194 and the Law on Inspection of Construction No.4708. In addition, there are various building codes prepared by the authority. The main ones are: Housing and Zoning Code, Fire Code, Shelter Code, Parking Code, Elevator Code, Codes for specific building uses (e.g. private hospitals, public housing, high-rise structures, construction in disaster areas)

Individual municipalities have their own housing and zoning codes that include the rules defined by the ministry documents and add further specifications. İzmir Municipality Housing and Zoning Code (IMHZcode) has been chosen for this study, as İzmir is the third most populous city in Turkey and IMHZcode is representative of codes that are in effect across Turkey. IMHZcode was analyzed to understand the characteristics of building codes and to develop a building code model. First, the organization of building code sections has been examined, then the basic elements of building codes have been identified, and after that rules in the building code have been classified according to their self-containedness and formalizability.

2.1 Building Code Organization

IMHZcode is the legal document that specifies minimum conditions that need to be satisfied by settlements and construction operations within the İzmir Metropolitan Municipality and its environs. Its main structure is formed with six different parts as follows:

Part I: General Rules (from Clause 1 to Clause 10)
Part II: Definitions (11-23)
- Chapter 1: Definitions of Municipal (Development/Zoning) Plans (11-16)
- Chapter 2: Definitions of Constructions (17-19)
- Chapter 3: Definitions of Construction Permit and Building Occupancy Permit (20-23)

Part III: Rules Related to Buildings and Land Readjustment (24-66)
Part IV: Rules Related to Construction Permit and Building Occupancy Permit (67-76)
Part V: Buildings, Building Parts and Facilities Subject to Special Rules (77-86)
Part VI: Rules Rescinded, Interim Provisions and Entry in to Force (87-89)

IMHZcode consists of 26 different clauses containing rules that are related to buildings (clause 40-clause 66) whereas the rest of the building code is informative or unrelated to buildings. Each of the clauses consists of several rules defines constraints relating to one specific object such as roofs, windows, doors, staircases etc.

2.2 Building Code Elements

The process of representing building codes as computable rule sets is not trivial due to the complex nature of building codes. It is essential to have an understanding of the various types of information contained in building codes as well as the organization of the documents in order to develop a building code model. Building codes consist of two different types of expressions: graphical and textual. Graphical expressions, such as figures, drawings, diagrams, and pictures are used in documents related to standards such as accessibility, parking and elevator standards. IMHZcode (İzmir Metropolitan Municipality, Housing and Zoning Codes) and most of the building codes in Turkey do not include graphical expressions and this research mainly concentrates on representation of textual expressions of building codes.

Three types of textual expressions exist in the building codes investigated:

1. General provisions
2. Definitions
3. Rules

General provisions describe the aim, scope and legal basis of the building code. For example, the following expression quoted from IMHZcode is a general provision that describes the scope of this code.

> " This code, prepared in accordance with Zoning Law No:3194 and rule 8 of Code for the Implementation of Law No: 3030 on Management of the Metropolitan Municipalities, is applied within the boundaries of and the contiguous area of İzmir Metropolitan Municipality."

Definition statements define the specific names used in the building code and give detailed descriptions about definitions. For example, the following expression quoted from IMHZcode defines meaning of a name (High-rise building).

> "A building, height of which > 30.80m or has more than thirteen (13) storeys."

Rule statements define constraints about physical building components, spaces and relations. For example, the following expression quoted from IMHZcode defines a constraint about roof eaves.

> "Roof eaves can be done throughout the entire building facades."

Building projects are checked against these rule statements for their compliance.

2.3 Rule Classification

Building codes are written text documents, to be interpreted by humans. They are not structured in a strict and straightforward manner that can be interpreted by machine. They have a complicated structure. While some simple rules can easily be defined in a single statement, others require a series of statements making exceptions, clarifications and modifications. The analysis of IMHZcode structure has revealed two types of rules:

- Self-contained rules
- Linked explanatory rules

Self-contained rules. They indicate how something will be, must be, should be, or can be. A rule related to the width of the stairs is an example of this type.

> "The minimum width for flights and landings of stairs shall be 1.20 m."

Linked-explanatory rules. These rules are clarifications, exceptions, exemptions, or modifications of other rules. For example, consider the following two rules from IMHZcode, one modification and one exception example, for the above rule on the minimum width for stairs.

> "These dimensions can be reduced to 0.90 m. for single-family house, basement, and service stairs."

> "These dimension restrictions may be ignored for stairs leading to attics that are not occupied."

In addition to a complex structure, building codes contain rules that may be open to interpretation, ambiguous and sometimes even contradictory and therefore impossible to model completely. Classification of rules according to their formalizability is necessary to assess the potential coverage of building code representations. Three additional types of rules have been identified:

- Formalizable rules,
- Semi-formalizable rules, and
- Non-formalizable rules.

Formalizable rules. They can be represented in a computer implementable format and allow for automated compliance checking without any ambiguities. The following is an example from IMHZcode:

> "Roof slope cannot exceed 33%."

Semi-formalizable rules. They contain fuzzy concepts that require interpretation (e.g. enough, easily, nearly, appropriate, and approximately). These rules require clarification of the concepts involved either during modeling of the rule or later during compliance checking. Example:

> "Spaces left as shelter must be able to dispose of garbage easily."

Non-formalizable rules. They rely on qualitative evaluations such as ones based on aesthetics or characteristics as well as evaluations where local authority is allowed to use initiative. These rules are impossible to represent in computable format and necessitate manual compliance checking under all conditions. Example:

> "Roofs must be compatible with the building and in harmony with the character of the streetscape. "

In the final stage of the analysis, IMHZcode is decomposed into a list of rules. 258 rules are found on residential buildings. All rules have been classified according to their formalizability and whether they are self-contained or not. For example, rules on residential buildings of clause 41 of IMHZcode are shown in Table 1.

Table 1: Classification of rules on residential buildings contained in clause 41 of IMHZcode

Clause 41- Roofs		
Rule statements	**Rule type**	
	Self-contained vs Linked-explanatory	**Formalizability**
Roofs must be compatible with the building and in harmony with the character of the streetscape.	Self-contained	Non-formalizable
Roof slope cannot exceed 33%.	Self-contained	Formalizable
The roof pitch is calculated from building façade without taking into consideration the eave's width.	Linked-explanatory (clarification)	Formalizable
Independent units cannot be built in the attic.	Self-contained	Formalizable
In these spaces only elevator towers, central air conditioning systems, and chimneys can be built.	Linked-explanatory (exception)	Formalizable
Storage for coal cannot be placed in the attic or flat roofs.	Self-contained	Formalizable
Skylights, forehead and gable walls, solar panels and tanks cannot rise more than 0.60 m. from the roof plane.	Self-contained	Formalizable
No extensions are allowed above the roof except for chimneys and airshafts.	Self-contained	Formalizable
If Masonry parapets' height is less than 1.10 m., it is not included in the building height	Linked-explanatory (clarification)	Formalizable

The classification of IMHZcode rules on residential buildings revealed that 58% of the rules are self-contained and formalizable and 21% are explicative and formalizable. As indicated in Table 2, 79% of IMHZcode rules on residential buildings can be represented in a computer implementable format.

Table 2: Results of the classification of IMHZcode rules on residential buildings

	Formalizable rules	**Semi-formalizable rules**	**Non-formalizable rules**
Self-contained rules	58% (149)	7% (17)	4% (12)
Linked explanatory rules	21% (55)	6% (14)	4% (11)
Total	79% (204)	13% (31)	8% (23)

3. Building Code Representation

3.1 Building Code Models

Through the analysis of IMHZcode, different types of rules to be represented have been identified. Afterwards, an investigation for the appropriate representation was carried out. In this step, existing representation approaches have been evaluated especially in terms of maintainability. Although there has been a lot of interest in building code modeling, there are still no standards or defined methods to represent building codes in computer implementable formats. The most common representation approach is to hardcode rules in computer programming languages. The main disadvantage to this approach is that it requires a high-level of expertise in computer programming to define, write and maintain building codes. However, the ability to update and maintain the representation is important because building codes change continuously.

Existing building code models are summarized below:

Basic Models (Decision Tables). Decision tables (Fenves, 1966) are the initial efforts on building code representations. In this model, building code rules are represented in the precise and unambiguous form of decision tables. However, each table represents only one rule and therefore, decision table-based models need interlinks among various tables because rules have often internal relationships between each other. Such interlinks among tables make working with decision table based models difficult.

Condition/Consequence Models. If-then conditions are defined for building code rules in condition/consequence models (Garrett and Fenves, 1987). These models are flexible and easily implementable in code compliance checking systems. However, this kind of models lack an organization of the building codes and requires many redundancies when defining **rules.**

Logic Models. Formal logics are used to represent building code rules in logic models (Hakim and Garrett, 1993; Rasdorf and Lakmazaheri, 1990). Logic models also use if-then conditions. Knowledge of logic is required to model building codes and understand them. Logic models include user-defined predicates and logical operators that prevent them from being widely implementable.

Object-Oriented Models. Object-oriented models (Garrett and Hakim, 1992) use classes and attributes to represent building code rules. Although object-oriented models have some advantages (e.g. flexibility, and extensibility) as compared with previous models, there are still some difficulties in maintainability. The main difficulty is that these models are only editable by users who have object-oriented programing knowledge. Moreover, object-oriented models are less human readable and understandable.

Hybrid Models. Hybrid models combine some of the previous representation approaches. Yabuki and Law (1993) combined object-oriented models and first order predicate logic to develop an Object-Logic model. Kiliccote (1994) developed a context-oriented model, which utilizes an object-oriented approach as well. This model addresses the same problem of the complexity of the previous object oriented models. While hybrid models benefit from advantages of the combined approaches, they inherit shortcomings from the previous approaches such as the problem of complexity.

Semantic Models (SMARTcodes). The SMARTcodes (AEC3, 2012) is a semantic approach which proposes to mark-up building codes in such a way that rules are automatically generated in a computer implementable format. This approach will be explained in detail in

the next section. Based on simple constructs that can be identified from actual text of the rules, it offers a higher level of maintainability by non-programmers and is found to be the most appropriate approach for representing IMHZcode.

3.2 SMARTcodes and RASE Methodology

SMARTcodes provides a protocol and a software program for creating tagged representations of actual building code texts (Conover, 2009). It is based on a process using a mark-up language to mark the actual text of the building code according to SMARTcodes protocol. This can be done by code authors. The mark-upped text is structured into an XML version of the actual building code. The structured XML is then converted into computer implementable rules. The SMARTcodes protocol defines which textual parts should have one of four colors, each representing one of the four constructs that rules are built on. The four constructs are defined by the RASE methodology. The main goal of the RASE methodology is to identify the common constructs for building code rules. It states that building code rules can be broken down into four constructs: Requirement, Applicability, Selection, and Exception.

Building codes contain a number of rules, and each rule contains of a number of requirement, applicability, selection, and exception parts. Every rule must have at least one **requirement** indicator. It is the condition that must be satisfied by one or more aspects of a building. Similarly, every rule must have at least one **applicability** indicator that defines which aspects of the building the requirements apply to. Applicability indicators can be seen as a definition of scope associated with the rule. Rules may have **selection** indicators if the rule is for specified cases among the applicable elements. Rules may also contain **exception** indicators. Exception information identifies the conditions under which the rule is not applicable to the building elements. RASE methodology utilizes these four types of indicators as a basis of the common constructs of rules. Each of these four constructs has attributes such as a property, a comparator and a target value with a unit (Nisbet et al., 2009). Code authors are able to markup these indicators that appear in the actual text of the code using SMARTcodes Builder software which creates and XML formatted version of the code. A screenshot of the SMARTcodes Builder software is shown in Figure 1.

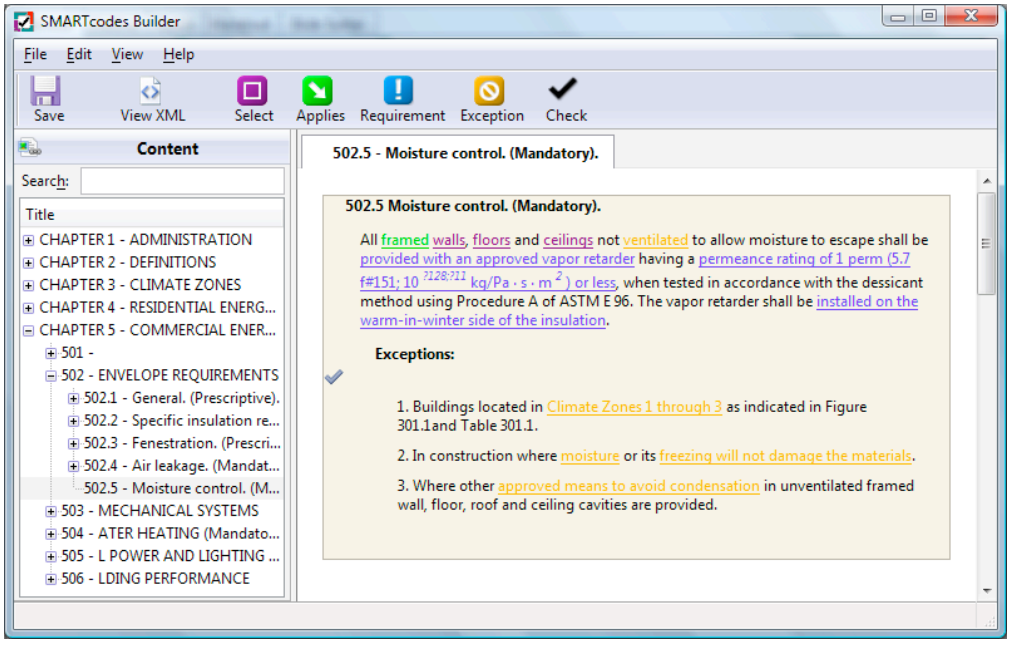

Figure 1: SMARTcodes Builder (AEC3, 2012)

3.3 Representation of IMHZcode

Among existing representation approaches, the RASE methodology has been found as the most appropriate for representing IMHZcode since it accommodates a scheme where code authors are able to maintain the representation.

IMHZcode rules have been modeled as requirement, applicability, selection, and exception objects. Each object is attributed to have an id, a property, a comparator, a target value, and a unit. Property is a term defined within the code. A value may be numeric, descriptive or boolean, while the comparators include greater, lesser, equal, etc. Some examples of how rules are modeled are shown in Table 3.

Table 3: Examples of rules from IMHZcode modeled according to the RASE method

	Rule text	**Applicabilities**	**Selections**	**Exceptions**	**Requirements**
1	Clear height of doors shall be at least 2.10 m.	door	-	-	door.height>= 2.10m
2	Clear width of entrance doors of independent unit shall be at least 1.00 m.	door	type=entrance	-	door.width>=1.00m
3	Buildings shall have at least one non-wood staircase.	building	-	-	hasStair=true & stair.material=!wood
4	The minimum width of a flight and a landing shall be 1.20 m.	stair	-	-	flightWidth=1.20m landingWidth=1.20m
5	Roofs in general must remain within 33% sloping height, except duplex houses.	roof	-	Building.type= duplex houses	pitch<=33%

Building code rules, in a RASE-based structured form, are stored in a relational database. Rules are entered using the database interface, however, the goal is to provide a simple tool for building code authors to enter rules. Acceptance of building code models by the AEC industry is not possible without tools directly usable by the building code authors who have no programming background. Building code authors, without assistance, should be able to create new rules and update existing ones.

The testing of the representation was done through an actual implementation. Currently, two commonly used compliance-checking applications are Express Data Manager (EDM), and Solibri Model Checker (SMC). EDM has a module for writing new rules in EXPRESS, but it is complex and requires a high level of expertise. SMC rules are hard coded into the system and SMC does not support adding new rules. As a result, neither of these systems is suitable as a test bed for IMHZcode representation. For this reason a new system has been implemented for compliance checking of building model with IMHZcode model. It consists of three main components. Building Code Rule Builder, Building Code View Builder, and Checker. Figure 2 illustrates a conceptual framework for compliance checking system.

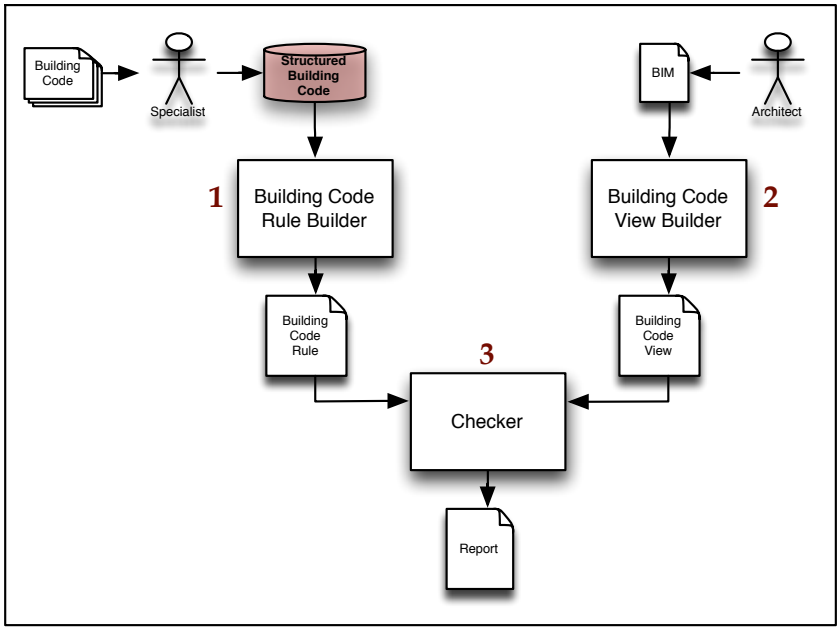

Figure 2: Conceptual framework for compliance checking system

Building Code Rule Builder connects to the database where structured building code rules are stored, reads the applicable structured rules and prepares rule instances for Checker. Building Code View Builder accesses and extracts the building model data and derives the appropriate model view for Checker. The third and major component is Checker. It applies rules to the building model and returns a report. A building element passes a check if it either is not applicable or is exempt or is as required. Figure 3 is a screenshot where compliance checking of a simple building with three rooms and three doors against door rules takes place.

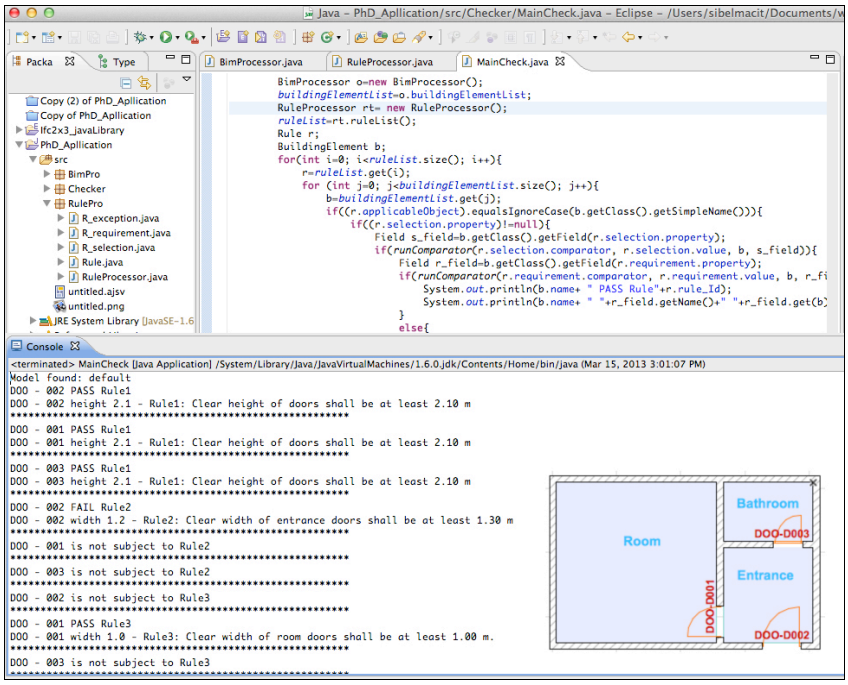

Figure 3: A screenshot of the testing application

4. Conclusion

Analyzing the complex structure of building codes and determining different types of rules is the first step in developing building code models. It is important to document how much of the building code and which types of rules can be modeled reliably in automated compliance checking systems. The analysis on the IMHZcode has identified 3 types of rules according to their formalizability. 79% of the rules are formalizable and are modeled easily. Another 13% are semi-formalizable. These can be modeled once the responsible authority resolves the ambiguities. Only 8% are non-formalizable and cannot be included in automated checking. Formalizable and semi-formalizable rules constitute 92% of the IMHZcode.

Building codes are updated continuously and the model needs to be maintained by the authority responsible for code checking. Therefore, evaluation of previous work on representations focused on maintainability. SMARTcodes approach with a protocol for converting existing codes from text to computer implementable format was chosen. SMARTcodes is based on RASE which is a simple method of structuring information embedded in the rules. The formalizable IMHZcode rules were successfully represented in computational format using RASE constructs. Yet, an approach to checking semi-formalizable rules still needs to be developed along with a process that includes how non-formalizable rules are handled.

5. Acknowledgements

This study was supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK).

References

AEC3. 2012. *International Code Council* [Online]. Available: http://www.aec3.com/en/5/5_013_ICC.htm [Accessed February, 23, 2013].

Conover, D. 2009. *Method and apparatus for automatically determining compliance with building regulations*. Washington, DC, US patent application 20090125283.

Fenves, S. J. (1966). Tabular Decision Logic for Structural Design. *Journal of Structural Division ASCE,* 92**,** Garrett, J. H. and Fenves, S. J. (1987). A knowledge-based standards processor for structural component design. *Engineering with Computers,* 2**,** 219-238.

Garrett, J. H. J. and Hakim, M. M. (1992). Object-Oriented Model of Engineering Design Standards. *Journal of Computing in Civil Engineering,* 6**,** 323-347.

Hakim, M. M. and Garrett, J. H. (1993). A description logic approach for representing engineering design standards. *Engineering with Computers,* 9**,** 108-124.

Han, C., Kunz, J. C. and Law, K. H. (2002). Compliance Analysis for Disabled Access. *In:* William J. McIver, J. a. A. K. E. (ed.) *Advances in Digital Government Technology, Human Factors, and Policy.* Kluwer, Boston, MA.

Kiliccote, H., James H. Garrett, J., Chmielenski, T. J. and Reed, K. A. (1994). The Context-Oriented Model: An Improved Modeling Approach for Representing and Processing Design Standards. *In:* Khozeimeh, K., ed. First ASCE Congress on Computing in Civil Engineering, June 1994, Washington, D.C. New York:ASCE, 145-152.

Nisbet, N., Wix, J. and Conover, D. (2009). The Future of Virtual Construction and Regulation Checking. *Virtual Futures for Design, Construction & Procurement.* Blackwell Publishing Ltd.

Rasdorf, W. J. and Lakmazaheri, S. (1990). Logic-Based Approach for Modeling Organization of Design Standards. *Journal of Computing in Civil Engineering,* 4**,** 102-123.

Yabuki, N. and Law, K. H. (1993). An Object-Logic model for the representation and processing of design standards. *Engineering with Computers,* 9**,** 133-159