A Hybrid Model for Building Code Representation Based on Four-Level and Semantic Modeling Approaches

Sibel Macit, sibelmacit@gmail.com
Balıkesir University, Turkey

M. Emre İlal, emreilal@iyte.edu.tr
Izmir Institute of Technology, Turkey

Georg Suter, georg.suter@tuwien.ac.at
Vienna University of Technology, Austria

H. Murat Günaydın, gunaydinh@itu.edu.tr
Istanbul Technical University, Turkey

Abstract
This paper presents a study in the field of automated compliance checking, concentrating on building code representations. A new model for representing building codes in computable form is developed for use in building automated compliance checking systems. The model adopts Nyman and Fenves's four-level representation paradigm as a theoretical base and uses the semantic modeling approach of the SMARTcodes project for developing the building code representation. This hybrid model breaks down the representation into four levels which allows separate modeling of domain concepts, individual rule statements, relationships between rules, and the overall organization of the building code.

The applicability of the model has been evaluated with a case study. The İzmir Municipality Housing and Zoning Code has been chosen as a document that represents a complex code document that is in effect throughout Turkey. The formalizable rules in this code have been modeled based on the new representation. This research shows that decomposing a building code into four levels and modeling rules based on the semantic-oriented paradigm is an effective modeling strategy for representing building codes in a computable form that is independent of automated compliance checking systems.

Keywords: Building code representation, automated compliance checking

1 Introduction

Automated code compliance checking has long been an area of research. Computational support for compliance checking of building designs against applicable codes promises accurate and time effective operations. However, complexities involved have long been acknowledged in literature and a working system still remains elusive (Fenves et al. 1995; Froese 2010). Since the industry wide adoption of Building Information Modeling (BIM) systems, the level of interoperability among various tools and processes has been gradually increasing. This promising development has renewed the interest in automated code compliance checking in recent years (Nisbet et al. 2009; Eastman et al. 2009).

Automated compliance checking systems are expected to retrieve a set of building codes from related authorities and conduct compliance checking on submitted building designs. Compliance checking systems primarily require appropriate computer-based models of both building codes and building designs. Advances in BIM tools have finally established a standardized representation for building designs, even if it is currently deemed unsatisfactory. However, a standard representation for building codes is still not available. The lack of a standard representation for building codes has inhibited the development and use of commercial automated compliance checking systems.

There has been extensive research conducted internationally over the last four decades in the area of representing building codes in a computable format for automated compliance checking. Several researchers have suggested various building code representation models and automated compliance checking environments based on these models (Rasdorf & Lakmazaheri 1990; Jain et al. 1989; Yabuki & Law 1993; Garrett & Hakim 1992; Kiliccote et al. 1994; Pauwels et al. 2011; Yurchyshyna & Zarli 2009; Fenves 1966; Nisbet et al. 2009; Eastman et al. 2009). However, there has been limited success in transferring these environments into practice (Fenves et al. 1995; Hakim & Garrett 1992). A literature survey reveals that the reasons for this failure are related with the building code models used in these environments and not with the specific implementations of these environments (Fenves et al. 1995). Previous building code models have several limitations. One limitation is being too simplistic and not comprehensive enough compared to the complex nature of building codes and thus lacking the capability to represent all, or almost all, of the various types of information in building codes. A second limitation is that some building codes are hard-coded into the systems, thus lacking flexibility, maintainability, and user control (i.e. non-programmer users cannot add/modify the rules embedded in the system and cannot make professional judgments on the model). Furthermore, any change in the building code necessitates changes in all such systems. A third limitation is that there is no direct mapping between the building codes and building code models, making consistency checking between actual building code and code model difficult. A fourth limitation is only focusing on individual rule representation ignoring the overall building code and thus lacking capability to prevent contradictions.

This paper presents a hybrid model establishing a methodology for building code representation that addresses the shortcomings identified in previous efforts. Ideally, the building code representation should be independent of compliance checking systems and be adaptable to continuous amendments to the building code. It should be consistent (preventing ambiguities as well as contradictions among rules) and comprehensive. Moreover, there should not be representational redundancies in the building code model. The hybrid model aims to meet the above requirements. It is intended to be utilized as a basis for the development of automated compliance checking systems.

This paper also presents a case study for building a representation of an actual building code. For the case study, İzmir Municipality Housing and Zoning Code (IMHZCode) has been chosen. IMHZCode is representative of codes that are in effect throughout Turkey. IMHZcode rules pertinent to residential buildings have been represented in computer implementable format based on the hybrid model.

2 Background

2.1 Four-level representation

The four level representation paradigm (Fenves & Wright 1977) has been adopted as a theoretical base for the new model. It is based on an abstract model of the logical structure of building codes identified by Nyman & Fenves (1975) who investigated possible methods of restructuring building codes. According to this general structure, the content of the building code is described in following four levels: 1) *The top level* provides the overall organization of the building code by grouping related statements into larger units (sections), 2) *the intermediate level*, i.e. the level of the sections defined by the top level, deals with a set of closely related statements and their dependency relationships (clauses), 3) *the detailed level*, i.e. the level of the clauses defined by the intermediate level, concerns individual statements (rules), and 4) *the lowest level* corresponds to the leaf nodes of the representation tree and describes the terms referred to in the statements (concepts) and allows mapping to standardized building information models.

Identifying the nature of building codes and the hierarchy of information in them is important for deciding on a modeling approach that can provide an effective method for the development of building code representations. Although Fenves and Nyman's work provides a solid theoretical foundation for representation of building codes, previous models based on this theory (Fenves et al. 1987; Jain et al. 1989; Rasdorf & Lakmazaheri 1990) were not widely adopted in AEC industry mainly due to the immature information technologies for knowledge representation at the time. Representation methods available for modeling of building code information were inefficient and confusing for non-programmer users, therefore code models quickly became hard to maintain and build. However, the four-level structure they introduced is still applicable and has proven to be a robust method for decomposing building codes.

The hybrid model adopts the four level paradigm but addresses the issues on knowledge representation methods by utilizing the relatively recent semantic modeling method that will be discussed in the next section.

2.2 Semantic representation

Previous building code representation research efforts mainly focused on the hard-coding approach. 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 building code representation is essential due to the fact that building codes change continuously. To overcome the deficiencies of hard-coded representation approaches, recently, attention has been directed towards the semantic modeling approach, which is a relatively new method for knowledge representation. One recent project, SMARTcodes, can be referenced as a good example of the use of this approach (Nisbet et al. 2009). SMARTcodes' semantic-oriented representation approach provides an easy to understand, elegant method for modeling rule statements. It utilizes a simple scheme (RASE model) that is applicable for all types of rule statements, and with it, non-programmer users are able to build and maintain building code representations (Hjelseth & Nisbet 2010).

RASE (Requirement, Applicability, Selection, Exception) model defines common constructs that make up a rule. It states that building codes contain a number of 'checks', which typically demarcate a distinct section of the building code, and each check is made up of a number of requirement, applicability, selection, and exception parts. According to this model, every check 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 check 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 check. Checks may have *selection* indicators if the rule is for specified cases among the applicable elements. Checks may also contain *exception* indicators. Exception information identifies the conditions under which the check is not applicable to building elements. RASE model utilizes these four types of indicators as a basis of the common constructs of checks and offers a scheme for code authors to build and maintain building code representations.

A pilot study carried out in the early phase of the research showed that RASE model can be utilized for representing real building codes, however, several shortcomings were identified (Macit et al. 2013). First, it is prone to inconsistencies and creates redundancies arising from modeling the same concepts multiple times for each applicability and selection construct when the concepts are referenced by multiple rules. In the SMARTcodes project, an external dictionary is used to prevent inconsistencies. Second shortcoming is the lack of explicit relationships between individual rule statements. In SMARTcodes' approach, relationship representation (hierarchy within rules) is handled separately by the automated compliance checking system. This makes it difficult to ensure correctness and consistency for the overall code representation, independent of automated checking systems.

The hybrid model proposed in this paper adopts the semantic modeling approach of the SMARTcodes project for representing building code rule statements but modifies it into a four level representation that improves it by eliminating redundancies and adding logical relationships. The hybrid model is discussed in detail in the next section.

3 Building Code Representation

The proposed building code representation combines the semantic representation method established by the SMARTcodes project with the theoretical foundations set by Nyman & Fenves (1975), namely the four level representation. This hybrid approach is proposed to provide a systematic structure for representing building codes in computer implementable format. The hybrid model organizes the representation in 4 levels (
Figure 1). The four levels are:

1. The *domain level* which models the concepts, which are mentioned in the original building code text, their attributes and relationships.
2. The *rule level* where individual rule statements of the building code are represented in a structured format, utilizing the concepts modeled at the domain level. The rules are modeled based on modified RASE constructs.
3. The *rule-set level* where relationships between rule objects are defined forming the rule-sets.
4. The *management level* which reflects the overall organization of the building code model by connecting and categorizing the rule-sets.

The four levels are discussed in the following sections starting with the lowest level.

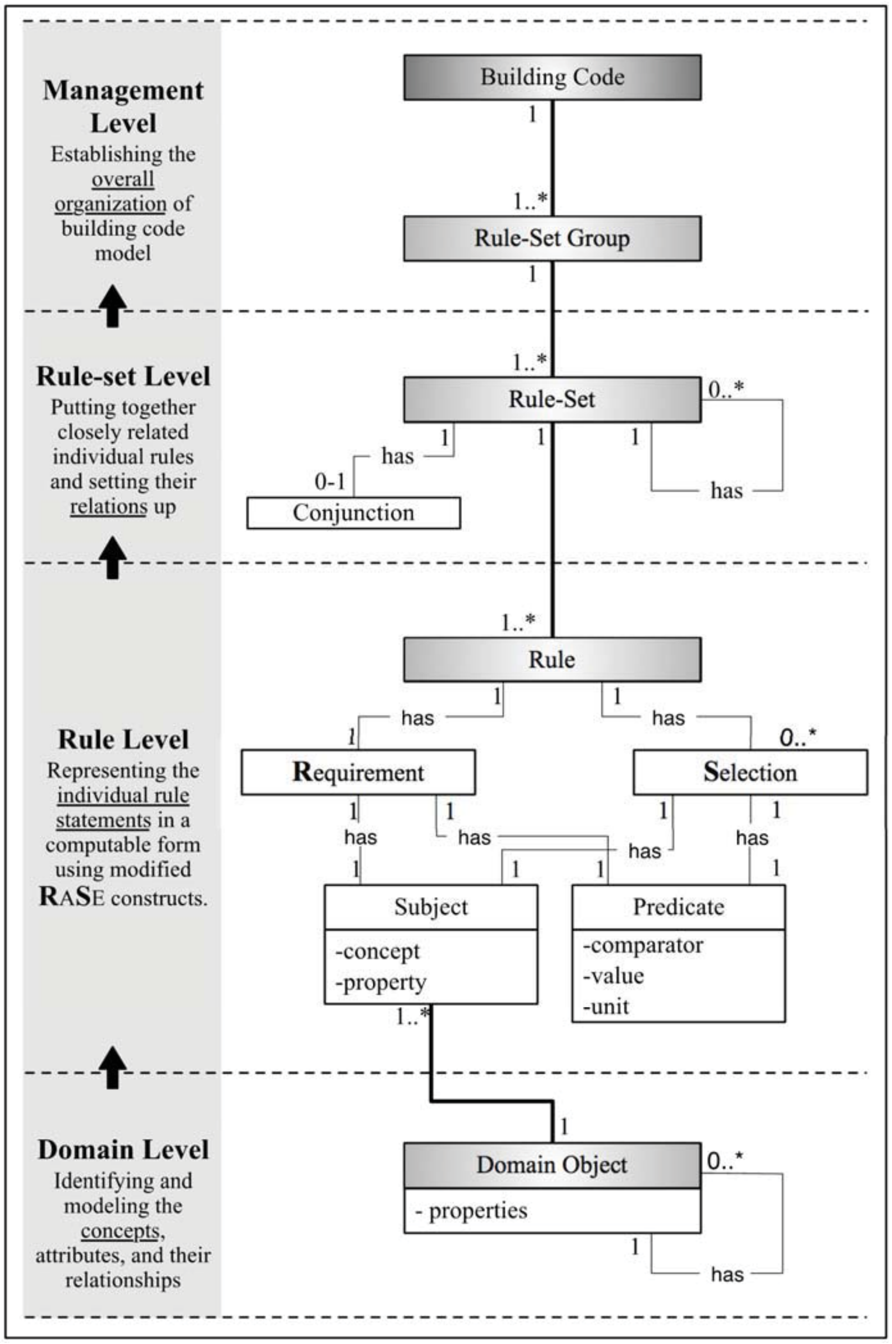
Management Level
Establishing the overall organization of building code model
Building Code
1
1..*
Rule-Set Group
1
Rule-set Level
Putting together closely related individual rules and setting their relations up
1..*
Rule-Set
0..*
1
1
1
has
0-1
Conjunction
has
Rule Level
Representing the individual rule statements in a computable form using modified RASE constructs.
1..*
Rule
1
1
has
has
1
0..*
Requirement
Selection
has
has
has
has
Subject
-concept
-property
Predicate
-comparator
-value
-unit
1..*
Domain Level
Identifying and modeling the concepts, attributes, and their relationships
1
Domain Object
- properties
0..*
1
has

Figure 1 Four-level structure of the hybrid model

3.1 Domain level

The lowest level of the hybrid model is the domain level for representing concepts and entities that appear in the building code document as object classes. Building codes refer to concepts specific to the domain which the codes are meant for (e.g. fire safety, accessibility) as well as entities that correspond to various aspects of the building project such as physical building components, spaces and relations (e.g. building, independent unit, storey, etc.). These domain specific concepts and entities constitute the domain level of the hybrid model. To create domain object classes, concepts and entities mentioned in the building code document are identified and modeled with their attributes and their relationships to each other.

The output of this level is a domain model that acts as a library of all required domain objects. The domain objects are utilized during the modeling of individual rule statements in the second level. The domain objects can be used as building blocks during modeling and maintenance of the rules by code authors with no programming background. This domain model is aimed at facilitating communication and interoperability among building information models, building code models, and automated compliance checking systems. The domain level also eliminates redundancies which occur in the modeling same concept many times by creating a lower level library.

3.2 Rule level

The second level of the hybrid model is the rule level for representing individual rule statements in computable format. Building codes include a set of rule statements that a building project must satisfy. Building projects are checked against the requirements and/or conditions, indicated by these rule statements. Automated compliance checking systems are rule-based systems and they require that rule statements are represented in a computable format. The individual rule statements modeled using modified RASE constructs form the rule level of the hybrid model. In the modified RASE, rules have only requirement and selection constructs. The applicability and exception information still exist. Applicability construct of the RASE model is embedded as the subject part of the requirement objects and it is filled by selecting from the library of domain objects. Exceptions are handled within the selection objects.

The output of this step is a rule model covering individual rule objects. Each rule object models a single requirement of the building code rule statements. In general, rule statements only have requirement information that indicates a quality requirement that must be satisfied by a domain concept. In some cases, rule statements also have selection information, if the requirement is for specified cases among applicable objects. Separate requirement and selection objects are modeled to capture this. Every rule object must have a single requirement construct and may have zero or more selection constructs. Both requirement and selection objects have the general form: A "subject" and a "predicate" (

Figure 1). The subject has a simple structure consisting of two basic elements: a concept, and a property (e.g., *door - height*). Concepts come from the *domain level* and may be a physical building component such as wall, door, slab, or an abstract concept such as space (living room), zone (independent unit). Properties are attributes of interest belonging to the concept. The predicates define the particular quality required of the subject (e.g., height of doors *shall be at least 210 cm*); each predicate has a comparator, a value, and a unit. The comparator is one of the relational operators (e.g. greater than, less than, equal to). The value is the specific value that is found in the code, whether numeric, descriptive, or Boolean. The unit simply specifies the unit of measure for the value.

3.3 Rule-set level

The third level of the hybrid model is the rule-set level for representing interrelations between the rule statements. Building codes are composed of various clauses that include closely related individual rule statements as well as the implicit or explicit information on the relationship among the statements. In the *rule level* each rule statement gets modeled with only one requirement for a specific property of a concept or entity and with selection information that clarifies the conditions under which the requirement applies to the concept or entity. However, in most cases an entity is subject to multiple requirements that vary according to the conditions. Multiple rule statements are used in order to specify and clarify conditions and requirements for a property of a concept or entity. Rules need to be connected representing the logical relationships that exist implicitly or explicitly within the semantics of a clause. Rules may be stand-alone, stating a requirement that is unrelated to other rules. However, for the most part, rules depend on each other. They either modify requirements or introduce additional requirements depending on the conditions.

In the hybrid model, related rule objects that are associated with the same subject are collected together into computable rule-sets by using logical conjunctions. AND conjunction is used for combining rule objects that indicate different values to be satisfied by a particular property of a concept simultaneously. OR conjunction is used for a relation between rules that indicate alternative values to be satisfied by a particular property of a concept depending on specified conditions. While all rule objects that are combined with an AND conjunction must be satisfied by the related concept, only one of the rule objects that are combined with an OR conjunction should be satisfied.

The output of this step is a collection of rule-set objects. This rule-set model is the logical combination of distinct rule objects. Rule objects are grouped into rule-sets, when they are all addressing the same subject (a property of a concept) that is being constrained. Rules are connected by logical conjunctions and form a tree where the root is the rule-set. The leaf nodes are the rules that have been modeled at the lower level.

3.4 Management level

The top level is the management level for representing the organization of the overall building code representation. Here, rule-set objects modeled in the third level can be categorized. Building codes are useful only if users (checkers or designers) can determine which portions of the building code pertain to their problem. To facilitate this, building codes are organized into chapters, sections, and paragraphs, with corresponding tables of contents and indexes. The user of a building code model should also be able to identify which rules of the building code apply for a given design situation. The building code model, therefore, needs to be organized in a systematic manner such that individual rules can be accessed easily. In the hybrid model, the *management level* addresses issues related to the organization of the rule-sets modeled in the *rule-set level.*

For categorizing rule-set objects, which reflect overall organization of the code representation, there are multiple methods. One natural method of categorizing rule-set objects is the one reflecting the structure of the original building code document. In this method, rule-set objects are grouped to reflect the clauses of the actual building code document. However, this method makes it difficult to recognize inconsistencies because of the scattered structure of building code documents. One alternative method that is appropriate for use by automated code compliance checking systems is grouping of rule-sets according to the concept they are related to. This allows automated checking to easily access all rules that apply to a given object. There may be many other possibilities in grouping rule-sets appropriate to the goal of the system being developed. It is possible to have multiple classifications existing independently at this level. The output of this level is the rule-set group objects, which will be employed by compliance checking algorithms to identify all needed rules to be processed for a given project.

4 Case Study

In order to provide a proof-of-concept implementation for the hybrid model, a case study has been conducted on modeling an actual building code. İzmir Municipality Housing and Zoning Code (IMHZCode) has been chosen as a code document for the case study since it is representative of complex codes that are in effect throughout Turkey. IMHZcode consists of 26 clauses containing 258 rule statements that are related to buildings and 79% of these rules is formalizable which can be represented in computer implementable format (Macit, 2013). In this case study, IMHZCode's all formalizable rule statements on buildings have been modeled based on the hybrid model. This implementation illustrates the process for representing an existing building code. The representation process consists of four steps for constructing a building code representation based on the hybrid model.

First step: Domain objects of the IMHZcode were created. For creating the IMHZCode domain objects, first, the IMHZCode was scanned manually, statement by statement, concepts and entities in the text are identified (e.g. building, story, space, door, etc.), and all related terms were extracted from it. After all terms were extracted, domain objects that represent the identified concepts and entities were determined and modeled as classes with required attributes and relationships to other classes. These classes, which are for utilization by multiple rule objects in the level above, along with the relationships between them form the domain model. The UML diagram for the resulting IMHZCode domain model is given in

Figure 2.

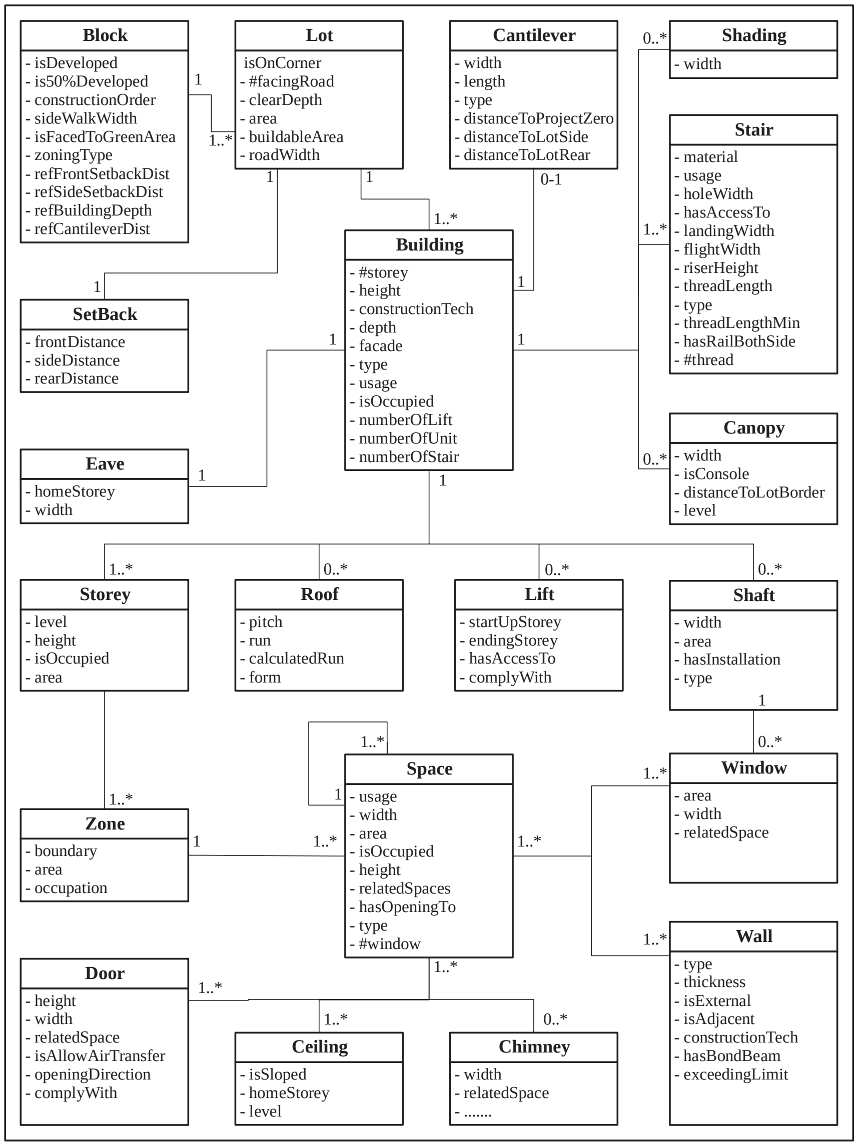

Figure 2 The domain model of IMHZCode

Second step: Individual rule statements in 26 clauses of the IMHZCode were structured using the developed rule model schema as a "semantic rule object". Each rule object has a "requirement" construct that describes the required specification in a concept. Some rule objects also have "selection" constructs describing the specific cases where the requirement is applicable. Both of these constructs have identical structures. They both have the following attributes: A concept, a property, a comparator, a value, and a unit. Table 1 shows the rule objects of Clause-27 as an example for illustrating how rule statements were modeled. Each rule object indicates a single requirement and is associated with a single property of a single concept defined in IMHZcode domain model. The relationships among these rule objects were modeled in the third step.

Table 1 The structured rule objects of IMHZCode Clause-27

Rule	REQUIREMENT					SELECTION			
Id	Concept	Property	C.	Value	U.	Concept	Property	Comp.	Value
R27.1	Setback	front Distance	≥	5	m				
R27.2	Setback	front Distance	=	{Block_refFrontSetbackDist}	m	Block Block	constructionOrder hasExistingBuilding	equal boolean	semiDetached true
R27.3	Setback	front Distance	=	{Block_refFrontSetbackDist}	m	Block Block	constructionOrder hasExistingBuilding	equal equal	plannedUnit true
R27.4	Setback	front Distance	=	{Block_refFrontSetbackDist}	m	Block Block	constructionOrder is50%Developed	equal equal	attached true
R27.5	Setback	side Distance	=	3	m				
R27.6	Setback	side Distance	=	(3+((:{Building_ numberofStorey}:-4)/2))	m	Building	numberofStorey	≥	4
R27.7	Setback	side Distance	≥	5	m	Building	constTechnique	equal	timberFramed
R27.8	Setback	rear Distance	=	(:{Building_height}:/2)	m				
R27.9	Setback	rear Distance	≥	3	m	Block	hasExistingBuilding	boolean	true
R27.10	Setback	rear Distance	=	{Block_refRearSetbackDist}	m	Block Block	constructionOrder hasExistingBuilding	equal boolean	semiDetached true
R27.11	Setback	rear Distance	=	{Block_refRearSetbackDist}	m	Block Block	constructionOrder hasExistingBuilding	equal boolean	plannedUnit true
R27.12	Setback	rear Distance	=	{Block_refRearSetbackDist}	m	Block Block	constructionOrder is50%Developed	equal boolean	attached true

Third step: Related rule objects that are associated with the same property of the same concept were collected together and modeled as nested rule-sets using one of the two logical conjunctions: AND, OR. Each top-level rule-set object is given an id, and defines the related concept and property associated with all rules in the set. A rule-set is defined for each concept property that is subject to a requirement in the code document even if there is a single rule object in the set. While some rule-sets have a simple one level relation between rule objects, some rule sets have multi-level (nested) relations. The rule-set objects of Clause-27 is given in Table 2 as an example.

Table 2 The rule-set objects of IMHZCode Clause-27

Id	Subject		Set
	Concept	Property	
RS27.A	Setback	frontDistance	(||: R27.1, R27.2, R27.3, R27.4)
RS27.B	Setback	sideDistance	(&: (||: R27.5, R27.6), R27.7)
RS27.C	Setback	rearDistance	(||: R27.8, (&: R27.9, (||: R27.10, R27.11, R27.12)))

Fourth step: The final fourth level (*management level*) of the hybrid model, allows for grouping of rule-sets. While a building project must simply be compliant with all rule-sets defined in the third level (*rule-set level*) regardless of how they are grouped, this level allows for modeling the structure of the code document itself as well as the relationships among rule-sets based on any aspect. In this case study, rule-sets were grouped based on the concept they are related to. In the IMHZCode there are instances where more than one clause is related with the same concept. For example, rules related with the *building* concept are distributed into three clauses. When checking the building objects for compliance it is more efficient to process all applicable rules for a class before moving on to other classes of objects. Table 3 illustrates classification of rule-sets of *setback* and *building* as an example.

Table 3 Classification of rule-sets related to setback and building concepts

Part	Concept	Property - Rule-set	Rule
III - Rules Related to Buildings and Land Readjustment			
	Setback		
		frontDistance –	[RS27.A = (||: R27.1, R27.2, R27.3, R27.4)]
			R27.1
			R27.2
			R27.3
			R27.4
		sideDistance –	[RS27.B = (&: (||: R27.5, R27.6), R27.7)]
			R27.5
			R27.6
			R27.7
		rearDistance –	[RS27.C = (||: R27.8, (&: R27.9, (||: R27.10, R27.11, R27.12)))]
			R27.8
			R27.9
			R27.10
			R27.11
			R27.12
	Building		
		depth –	[RS28=(||:R28.1,(&:R28.2,(||:R28.3,R28.4,R28.5)),R28.6, R28.7)]
			R28.1
			R28.2
			R28.3
			R28.4
			R28.5
			R28.6
			R28.8
		façade –	[RS29 = (R29.1)]
			R29.1
		height –	[RS30 = (||: R30.1, R30.2, R30.3, R30.4, R30.5, R30.6, R30.7, R30.8, R30.9, R30.10)]
			R30.1
			R30.2
			R30.3
			R30.4
			R30.5
			R30.6
			R30.7
			R30.8
			R30.9
			R30.10

5 Conclusion

This paper presented a hybrid model for the representation of building codes that may be utilized in the development of automated compliance checking systems. The hybrid model is the outcome of modifying and extending the recently developed semantic-oriented representation approach based on the theoretical view of the logical structure of building codes established by earlier efforts. The hybrid model divides the representation into four levels and allows to separate, representing domain concepts, individual rule statements, relationships between rules, and alternative methods of structuring (organizing) the overall building code document.

Decomposing a building code into four levels and modeling rules based on the semantic-oriented paradigm can be considered to be an effective modeling strategy for representing building codes in a computable format independent of automated compliance checking systems, since: i) This hybrid approach is able to preserve the high level of maintainability offered by the RASE model. Concepts, individual rule statements, relations between the rule statements and organization of the building code are separately represented. Required changes to the representation due to future revisions of the building code can be localized and handled without affecting the automated checking system. ii) The hybrid model minimizes redundancies that occur in the RASE model by introducing a hierarchical structure across four levels and improves on conciseness. iii) The hybrid model ensures consistency for building code representation by offering a level in which rule relationships are modeled and monitored independent of the automated checking system.

The case study on modeling IMHZcode demonstrates that the hybrid model is capable of representing building codes in a computer implementable format. However, there are limitations to this proof of concept. First, only IMHZCode has been modeled. Yet, it should be noted that IMHZCode is actually in effect and belongs to a metropolis. It is comprehensive and includes rule statements that have a high level of complexity. Thus, it is representative of codes that are hardest to represent in computational format. A second important limitation is the fact the hybrid model is designed with the intent to provide a solution for representing only formalizable rules. Semi-formalizable rules, which contain ambiguous or fuzzy concepts that require human interpretation, and non-formalizable rules, which rely on qualitative evaluations, have not been included in the scope of this research. As future efforts develop solutions for these types of rules, the methodology might need to be adapted.

References

Eastman, C. M., Lee, J.-m., Jeong, Y.-s. & Lee, J.-k. (2009). Automatic rule-based checking of building designs. *Automation in Construction,* 18**,** 1011-1033.

Fenves, S. J. (1966). Tabular Decision Logic for Structural Design. *Journal of Structural Division ASCE,* 92**,** 473-490.

Fenves, S. J., Garrett, J. H., Kiliccote, H., Law, K. H. & Reed, K. A. (1995). Computer representations of design standards and building codes: U.S. Perspective. *The International Journal of Construction Information Technology,* 3**,** 13-34.

Fenves, S. J. & Wright, R. N. (1977). The Representation and Use of Design Specifications. *NBS technical note 940.* Washington, DC.: National Bureau of Standards.

Fenves, S. J., Wright, R. N., Stahl, F. I. & Reed, K. A. (1987). Introduction to SASE: Standards Analysis, Synthesis and Expression. *In:* NBSIR (ed.). Washington, D.C.: National Bureau of Standards.

Froese, T. M. (2010). The impact of emerging information technology on project management for construction. *Automation in Construction,* 19**,** 531-538.

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

Hakim, M. M. & Garrett, J. H. (1992). Issues in modelling and processing design standards. The joint CIB Workshops on Computers and Information in Construction. CIB Publication 165, 242-257.

Hjelseth, E. & Nisbet, N. (2010). Exploring semantic based model checking. Applications of IT in the AEC Industry Proc. of the 27th International Conference CIB W78, 16-18 November, Cairo, Egypt. 16-18.

Jain, D., Law, K. H. & Krawinkler, H. (1989). On processing standards with predicate calculus. *In:* Barnwell, T. D., ed. Sixth Conf. on Computing in Civil Engineering, Atlanta, Georgia. Newyork: ASCE, 259-266.

Kiliccote, H., James H. Garrett, J., Chmielenski, T. J. & 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.

Macit, S., İlal, M. E., Günaydın, H. M. & Suter, G. (2013). İzmir municipality housing and zoning code analysis and representation for compliance checking. *In:* Suter, G., Rafiq, Y. and deWilde, P., eds. 20th Workshop of the European Group for Intelligent Computing in Engineering, 1-3 July, Vienna, Austria. EG-ICE.

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

Nyman, D. J. & Fenves, S. J. (1975). An Organization Model for Design Specifications. *Journal of structural Division ASCE,* 101**,** 697-716.

Pauwels, P., Deursen, D. V., Verstraeten, R., Roo, J. D., Meyer, R. D., Walle, R. V. d. & Campenhout, J. V. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction,* 20**,** 506-518.

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

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

Yurchyshyna, A. & Zarli, A. (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction,* 18**,** 1084-1098.